IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2011

EEE/ISE PART III/IV: MEng, BEng and ACGI

# ARTIFICIAL INTELLIGENCE

Monday, 16 May 10:00 am

Time allowed: 3:00 hours

**There are SIX questions on this paper.**

**Answer FOUR questions.**

*All questions carry equal marks*

**Any special instructions for invigilators and information for candidates are on page 1.**

| | | |
|---|---|---|
| Examiners responsible | First Marker(s) : | J.V. Pitt |
| | Second Marker(s) : | Y.K. Demiris |

## The Questions

1 a) Two parents with their triplets are on a beach and standing at the bottom of a cliff. There is no way round the cliff and they want to get to the top before the tide comes in. There is a basket attached to a rope and pulley that can be used to lift people up and down the cliff. The basket is big enough to fit all 5 people at once, but the rope is only strong enough to hold either 1 adult or 2 of the 3 children at a time.

Represent, in Prolog or other declarative notation, this planning problem, such that it could be used with the General Graph Search (GGS) program to compute a plan to get the family safely to the top.

[8]

b) A bin-emptying robot can be at one of four locations that are arranged in a square. Adjacent rooms are accessible, one from the other, via an adjoining door, which may be open or closed. Suppose the bin-emptying robot is a generalist robot and can empty bins, and can also open and close doors.

Represent, in Prolog or other declarative notation, this planning problem, such that it could be used with the General Graph Search (GGS) program to compute a plan to ensure all the bins are empty.

[8]

c) Suppose there was another robot, call it spousebot, that monitors the actions of the bin-emptying robot and obsessively tells it to close the doors behind it.

If the bin-emptying robot wants to avoid a nagging, briefly explain how it could change the state representation, and the search algorithm of choice.

[4]

2 a) A graph $G$ can be (explicitly) defined by a 3-tuple $G = <N, E, R>$, where $N$ is a set of nodes, $E$ is a set of edges, and $R$ is the incidence relation between nodes and edges such that if $(n_1, e, n_2) \in R$, then there is a link between the two nodes $n_1$ and $n_2$ whose (actual) cost is $e$.

A rooted graph has a unique node $s \in N$ from which all paths originate.

(i) Give an (explicit) definition of the paths in a rooted graph, and their costs.

(ii) Give an alternative (inductive) definition of the paths in a rooted graph, and their costs. Show that the paths in the alternative definition are the same, clearly stating any assumptions you make.

(iii) Explain how the General Graph Search (GGS) program explores the paths induced by the alternative definition, if it were using the uniform cost search algorithm.

[12]

b) In the context of A* search, define what is meant by the terms *heuristic*, *admissible heuristic*, *monotonic function*, and *pathmax equation*.

Explain why A* search is optimal and complete.

Briefly comment on the time/space complexity of A* search.

[8]

3 a) Briefly describe the differences between the Minimax algorithm and the Alphabeta algorithm for searching a state-space for a two-player game. State the main advantage of the Alphabeta algorithm.

[4]

b) Consider the search trees for a two-player game shown in Figure 3.1.

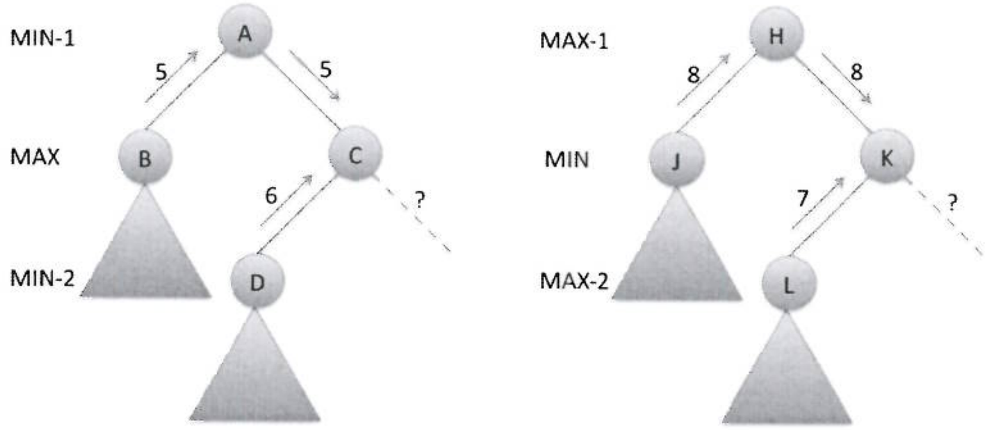

*Figure 3.1: Search Trees for a Two-Player Game*

Using the Alphabeta algorithm, explain whether or not the right-hand branches (dotted lines labelled '?') would be pruned.

[6]

c) Briefly describe the Reinforcement Learning algorithm for path-finding, e.g. in a grid-like maze.

Consider the grid maze shown in Figure 3.2, and a robot that can reliably sense a wall if it is front, behind, to its left, or to its right. Assuming a 'follow right wall' exploration strategy, show the map constructed after the robot has found the goal location G, and applied the Reinforcement Learning algorithm.

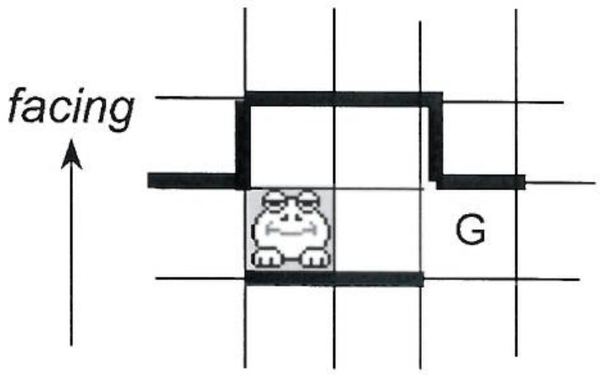

*Figure 3.2: Robot in a Grid Maze*

[6]

d) Explain how a robot could use the General Graph Search program with reinforcement learning to explore a maze and learn a path.

[4]

4 a) Describe a procedure for converting a set of first-order formulas into clausal form, i.e. where every conjunct is of the form:

$$\neg a1 \vee \neg a2 \vee \ldots \vee \neg am \vee b1 \vee b2 \vee \ldots \vee bn$$

[4]

b) What is significant about the clausal form if:

(i) $m = 0$ and $n = 1$;

(ii) $m > 0$ and $n = 1$;

(iii) $m > 0$ and $n = 0$;

(iv) for all the clauses in a set, $m \geq 0$ and $n = 1$.

[4]

c) The ISN research group has a saying that:

*If it looks like a thesis and it smells like a thesis, then it is a thesis.*

It is also rumoured that:

*If it passes the flick test, then it looks like a thesis.*
*If it passes the sniff test, then it smells like a thesis.*
*If one document passes the sniff test, and a second document weighs the same as the first document, then the second document passes the sniff test.*

Express these four statements as formulas of First Order Predicate Logic, and translate them into clausal form.

[4]

d) Consider the following facts:

*Brendan's document passes the flick test.*[1]
*Lloyd's document passes the sniff test.*[2]
*Lloyd's document weighs the same as Brendan's document.*

Express these facts in clausal form.

[2]

e) Prove, using resolution and showing the unifiers, that *Brendan's document is a thesis.*

[6]

[1] A document passes the flick test if the supervisor riffles through the pages and observes an adequate number of equations, graphs and fivers stapled to the pages.

[2] A document passes the sniff test if the supervisor opens it at a random page and inhales that unique aroma of printer ink, debt and desperation.

5 a) Given an inference system $I$, and letting $S$ be a set of formulas and $p$ a proposition:

(i) Explain what is meant by the meta-symbols: $\models$ and $\vdash_I$;

(ii) Explain the significance of the statement: $S \models p$ *if and only if* $S \vdash_I p$

[4]

b) Explain the relationship between disjunctive normal form and *KE* tableaux.

Explain the relationship between the Deduction Theorem and the proof method for *KE* tableaux.

[4]

c) Using the *KE* proof procedure, prove the following theorems.

(i) $\{p \vee q, p \rightarrow r, q \rightarrow r\} \vdash_{KE} r$

(ii) $\{p \wedge \neg p\} \vdash_{KE} z$

(iii) $\{(p \rightarrow p) \rightarrow q\} \vdash_{KE} q$

[6]

d) Suppose that a software agent has the following sentences in its knowledge base *KB*:

$KB = \{\neg(p \wedge \neg q) \vee \neg(\neg s \wedge \neg t), \neg(t \vee q), u \rightarrow (\neg t \rightarrow (\neg s \wedge p))\}$

Using the *KE* proof procedure, show $KB \models (\neg u)$.

[6]

6 a) Specify a syntax for well-formed formulas of propositional modal logic. Define a *Kripke model*, and specify how a Kripke model is used to give a semantics for formulas of modal logic.

[6]

b) Consider the following axiom schema. State the frame condition for the corresponding class of models:

(i) $\Box p \rightarrow p$

(ii) $p \rightarrow \Box\Diamond p$

(iii) $\Box p \rightarrow \Box\Box p$

(iv) $\Box p \rightarrow \Diamond p$

[4]

c) For *one* of the axiom schema defined in part (b), show that it does not hold in the class of all models.

Show that the same axiom schema does hold in the class of all models in which the accessibility relation satisfies the corresponding relation.

[4]

d) Prove, using the KE calculus for propositional modal logic, that the following axiom schema are all theorems of modal logic S5:

(i) $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$

(ii) $\Box p \rightarrow p$

(iii) $p \rightarrow \Box\Diamond p$

(iv) $\Box p \rightarrow \Box\Box p$

(v) $\Box p \rightarrow \Diamond p$

(vi) $\Diamond p \rightarrow \Box\Diamond p$

[6]

# ARTIFICIAL INTELLIGENCE SOLUTIONS 2011

1 (a) application
state representation ( location, (mother, father, triplets) )

where
location \in {bottom, top} is the location of the basket
mother \in {bottom, top} is the location of one parent
father \in {bottom, top} is the location of the other parent
triplets \in {1, 2, 3} is how many of the children are at the bottom of the cliff

start state = ( bottom, (bottom, bottom, 3) )

goal state = ( _, (top, top, 0) )

State change – only allow 'safe' actions. An alternative is to represent each person, and then have an additional state change as a move into the basket, and check that the backet occupancy is safe.

```
%move first adult up or down
state_change( ma, (Basket, (Basket, Other, N)) , (Opposite, (Opposite, Other, N)) :-
        opposite( Basket, Opposite ).

%move first adult up or down
state_change( pa, (Basket, (Other, Basket, N)) , (Opposite, (Other, Opposite, N)) :-
        opposite( Basket, Opposite ).

%move children up
state_change( bottom, (Basket, (Other, Basket, N)) , (top, (Other, Opposite, M)) :-
        (M is N – 1 ; M is N – 2 ).

%move children down
state_change( top, (Basket, (Other, Basket, N)) , (bottom, (Other, Opposite, M)) :-
        (M is N + 1 ; M is N + 2 ).

opposite( bottom, top ).
opposite( top, bottom ).
```

[8]

(b) application
state representation ( location, bins, doors )

where
location \in {0,1,2,3}
bins is a list of 4 elements, each element \in {full, empty}
doors is a list of 4 element, each element \in {open, shut}

initial state = derive from diagram

goal state = ( _, [empty, empty, empty, empty], _ )

```
% move
state_change( move, (LocOld, Bins, Doors), (LocNew, Bins, Doors) ) :-
        connected( LocOld, LocNew, Door ),
        append( Front, [open|_], Doors ),
        length( Front, Door ).

% open
state_change( open, (Loc, Bins, OldDoors), (Loc, Bins, NewDoors) ) :-
        connected( Loc, _, Door ),
        append( Front, [shut|Back], OldDoors ),
        length( Front, [open|Back], NewDoors).

%empty
state_change( empty, (Loc, BinOld, Doors), (Loc, BinNew, Doors) ) :-
        append( Front, [full|Back], BinOld ),
        append( Front, [empty|Back], BinNew ),
        length( Front, Loc ).

connected( 0, 1, 0 ).
connected( 1, 0, 0 ).
Etc.
```

(c)
Need a state-change-rule for close action

Need to record the path as part of the state

To check for a goal state, check that each move is followed by a close

Don't use depth first search as infinite loops of open-close action sequences

[4]

Grand total [20]

2
(a) [bookwork/understanding]
(i)
The explicit definition is given by:

Paths(G) = $_{i=0}\bigcup^{\infty}$ Pi
Where
P0 = { (<s>,0) }

Pi+1 = {(p ++ < $n_{i+1}$>, (cost+e)) | $\exists$($p_i$,cost) $\in$ Pi. frontier($p_i$) = ($n_i$) & ($n_i$,e,$n_{i+1}$) $\in$ R }

Since P'0 = P0, then every P'i+1 = Pi+1, on

(ii)
Alternative definition
G' = < start_node, Op > where
  start_node is the root node
  op is set of state transformers op: node -> node x edge
where the edge is the cost

Assume op computes all and only the members of incidence relation

Then the inductive definition of the paths and their costs in the graph is given by:

$P'_G = {}_{i=0}\bigcup^{\infty} P'_i$
where
  $P'_0$ = { (<s>, 0) }
  $P'_{i+1}$ = { ($p_i$ ++ <$n_{i+1}$>, (cost+e)) |
    $\exists$ op $\in$ Op . $\exists$ ($p_i$,cost) $\in$ $P'_i$ . ($n_{i+1}$,e) = op(frontier($p_i$)) }

Clearly P0 = P'0. Then P'i = Pi for all i because frontier($p_i$) = ni and op(ni) = ($n_{i+1}$,e) if and only if ($n_i$,e,$n_{i+1}$) $\in$ R

(iii)
Uniform cost first picks cheapest member of P'0. There is only one path

Therefore it generates every member of P'1.

Then it picks the cheapest member of P'1.

Therefore it generates a subset of P'2.

Then it picks the cheapest member of P'1 U (subset of P'2)

Then if it picks a member of P'1, it generates another subset of P'2 or
It picks a member of P'2, and it generates a subset of P'3

Then it either picks cheapest member of P'1 U (bigger subset of P'2) or
It picks cheapest member of P'1 U (subset of P'2) U (subset of P'3)

And so on.

Total [12]

(b)

heuristic: function which estimates cost of getting from current state to goal state
admissible: never over-estimates

Monotonic function: f-cost never decreases along any path in search space

Pathmax equation: forces heuristic admissible if

Optimality:
Optimal solution has cost f* to get to optimal goal G
Suppose A* search returns path to sub-optimal goal G'
We show that this is impossible

$$\begin{aligned} f(G') &= g(G') + h(G') \\ &= g(G') + 0 \qquad \text{G' is a goal state, we require h to be 0} \\ &= g(G') \end{aligned}$$

If G' is sub-optimal then $g(G') > f^*$
Now consider a node n on path to optimal solution G

```
        |-          n0 (start)
        |          /  \
        |         /    \
      f*|        n       G'
        |       /
        |      /
        |-   G
```

| Then: | | | | |
|---|---|---|---|---|
| | $f^*$ | $\geq$ | $f(n)$ | monotonicity |
| | $f(n)$ | $\geq$ | $f(G')$ | otherwise A* expands n first |
| | $f^*$ | $\geq$ | $f(G')$ | transitivity of $\geq$ |
| | $f^*$ | $\geq$ | $g(G')$ | a contradiction |

So either G' was optimal or A* does not return a sub-optimal solution.

Completeness:
A* expands nodes in order of increasing f-cost
Each expansion has lower bound > 0
So A* must eventually expand all nodes n with f(n) less than or equal to f*, one of which must be a goal state
(unless there are an infinite number of nodes with $f(n) < f^*$, or infinite number of nodes with finite total cost

Worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path), but is polynomial when the search space is a tree, there is a single goal state, and the error in the heuristic function h grows no faster than the log of the perfect heuristic (ie the heuristic that always computes the exact cost to the goal).

[8]

Grand Total [20]

3
(a) [bookwork/application]
minimax: exhaustively search
alphabeta: depth first to fixed ply

minimax: assign win or lose to leaf nodes
alphabeta: apply heuristic to evaluate utility of node

minimax: propagate 1/0 values up tree
alphabeta: offer parent to grand-parent value as alpha/beta cut-off

minimax:
alphabeta: offer alpha/beta cut-off value to other siblings to prune tree

Alphabeta prunes search tree which with a 'good' ordering of search can double the depth to which a search can be computer for the same time/space constraints.

(b)
B returns 5 as its candidate MIN-1 value
A offer 5 to C as its beta-cutoff
D returns 6 as its candidate MAX value
at Node C, alpha value 5 > beta cut-off
so 6 will lose to 5 going from Max -> Min-1 level
any branch on the rhs under ?: if it returns < 6, it will lose at min-2 -> max level
if it returns > 6, it will (also) lose at max -> min-1 level

J returns 8 as its candidate MAX-1 value
H offer 8 to K as its alpha-cutoff
L returns 7 as its candidate MIN value
at Node K, beta value 7 < alpha cut-off
so 7 will lose to 8 going from Min -> Max-1 level
any branch on the rhs under ?: if it returns > 7, it will lose at max-2 -> min level
if it returns < 7, it will (also) lose at min -> max-1 level

(c)
- **adopt basic strategy**
  - e.g. random walk, follow left wall, …, with some mechanism for loop checking
- **initialise** (starting grid location)
  - assign a coordinate value (say (0,0))
  - set reward to 0.0 and visited to true
  - perceive and record the result of performing action A (go up, down, left or right)
    - update pointers, set rewards to 0.0, visited to false, assign relative co-ordinate
- **explore**
  - choose next direction to go in and move one grid location in that direction
  - set visited for to 'true' for that grid location
  - perceive and record the result of performing action A (go up, down, left or right)
    - update pointers, set rewards to 0.0, visited to false, assign relative co-ordinate
  - repeat until exit location is found

➢ **assign** (reward or credit assignment)

➠ set reward of exit location, when found, to 10

➠ set reward of each visited grid location to 0.9 * maximum reward of any grid location that can be moved to by doing action A in that location

➠ propagate values back until all visited grid locations are assigned a reward (with loop checking…)

| location | north | south | east | west | visited | value |
|---|---|---|---|---|---|---|
| 0,0 | 0,1 | nil | 1,0 | -1,0 | true | 0.81 |
| -1,0 | nil | nil | nil | nil | false | 0.0 |
| 1,0 | 1,1 | nil | 2,0 | 0,0 | true | 0.9 |
| 0,1 | nil | nil | nil | nil | false | 0.0 |
| 1,1 | nil | nil | nil | nil | false | 0.0 |
| 2,0 (G) | nil | 2,-1 | 3,0 | 1,0 | true | 10.0 |
| 2,-1 | nil | nil | nil | nil | false | 0.0 |
| 3,0 | nil | nil | nil | nil | false | 0.0 |

(d)

GGS: node representation need to include state and reward.
State change operators –four 'action's, one to sense in each direction
When new paths are created, the new frontier node must be a variable X,
And the old frontier node must be unified with 0.9 * X (ie Y = 0.9 * X)
Goal state gives final frontier node value of 10.
Evaluate expressions to give reward for each node on the path.
Pass back search path and the rest of the graph (candidate paths)
Pick one of the candidate paths in the rest of the graph
Search from there (it is a partial graph with one path, and instead of being a path with one node (the start node) it will have several nodes, but otherwise GGS will work the same)
If you find a goal state, add that to your list of paths ending in goal states, add rest of the paths to the list of candidate paths.
This give explore, follow behaviour

[8]

Grand Total [20]

4
(a) [bookwork]
eliminate implication and equivalence
reduce scope of negation
rename variables
move quantifiers left without changing order
skolemize
drop universal quantifiers
convert to conjunctive normal form
make each conjunct a separate clause
give variables with same name in different clauses, different names

Total [4]

(b) bookwork/understanding
(i) fact
(ii) horn clause
(iii) goal/query
(iv) prolog program

Total [4]

(c) [application]

$\forall x \,.\, looks(x) \wedge smells(x) \rightarrow thesis(x)$
$\forall x \,.\, flicktest(x) \rightarrow looks(x)$
$\forall x \,.\, snifftest(x) \rightarrow smells(x)$
$\forall x \,.\, snifftest(x) \wedge weighssame(x, y) \rightarrow snifftest(y)$

$\neg looks(X1) \vee \neg smells(X1) \vee thesis(X1)$
$\neg flicktest(X2) \vee looks(X2)$
$\neg snifftest(X3) \vee smells(X3)$
$\neg snifftest(X4) \vee \neg weighssame(X4, Y1) \vee snifftest(Y1)$

[4]

(*d*) [*understanding/application*]

*flicktest*(*brendan*)
*snifftest*(*lloyd*)
*weighssame*(*lloyd*, *brendan*)

[2]

(*e*) [*application*]

*prove thesis*(*brendan*)

*start with negated conclusion* (*proof by refutation*)

$\neg thesis(\ brendan\ )$
$\neg looks(brendan) \vee \neg smells(brendan)\ \{X1 = brendan\}$
$\neg flicktest(brendan) \vee \neg smells(brendan)\ \{X1 = brendan, X2 = brendan\}$
$\neg smells(brendan)\ \{X1 = brendan, X2 = brendan\}$
$\neg snifftest(brendan)\ \{X1 = brendan, X2 = brendan, X3=brendan\}$
$\neg snifftest(X4) \vee \neg weighssame(X4, brendan)\ \{X1 = brendan, X2 = brendan, X3=brendan, Y1 = brendan\}$
$\neg weighssame(lloyd, brendan)\ \{X1 = brendan, X2 = brendan, X3=brendan, Y1 = Brendan, X4 = lloyd\}$
*nil*

[6]
Grand Total [20]

5

(a) [bookwork/understanding]

First off, $\models$ means S entails p: any valuation (assignment of truth values to propositional symbols) which makes every formula in S true, also makes p true

Then, $\vdash_I$ means S proves p in inference system I: starting from S, there is a sequence of formulas, where each formula follow from the previous one(s) by a rule of inference of KE, and ending in p (strictly speaking, we prove |-I (S -> p) and we use refutation, but the |- relation is the same)

Then

(i) is completeness: if there is an entailment, then I can prove it, i.e. |= implies $\vdash_I$

(ii) is soundness: if I proves something, then it is an entailment, i.e. $\vdash_I$ implies |=

Total [4]

(b) [bookwork]

➠ *KE proof procedure*

- *to prove $Q$, begin with $\neg Q$ and search for a refutation*
- *expand $\neg Q$ according to, but clearing away, logical structure of $Q$*
- *expansion takes form of tree, called a tableau, with formulas labelling nodes*
- *tableau is representation of disjunctive normal form*
  - *each branch represents the conjunction of formulas on the branch*
  - *the tableau is a disjunction of its branches*

*deduction theorem $S \cup \{A\} \vdash B$ implies $S \vdash A \rightarrow B$*

*So asked to prove $S \vdash_{ke} p$*

*Derive Sdash as the distributed and of the formulas in S*

*So by deduction theorem this is the same as proving $\vdash$ Sdash $\rightarrow p$*

*So $Q$ = (Sdash $\rightarrow P$)*

*So start from $\neg Q$ $\neg$(Sdash $\rightarrow P$)*

*By KE rules this amounts to writing down a line for each formula in Sdash and negating the conclusion*

Total [4]

c) [application]

(i)

| | | |
|---|---|---|
| 1 | $p \vee q$ | *premise* |
| 2 | $p \rightarrow r$ | *premise* |
| 3 | $q \rightarrow r$ | *premise* |
| 4 | $\neg r$ | *negated conclusion* |
| 5 | $\neg p$ | *beta* 2 4 |
| 7 | $\neg q$ | *beta* 3 4 |
| 8 | $q$ | *beta* 1 5 |
| | $X$ | 7 8 |

(*ii*)

| | | |
|---|---|---|
| 1 | $p \wedge \neg p$ | *premise* |
| 2 | $\neg z$ | *negated conclusion* |
| 3 | $p$ | *alpha* 1 |
| 4 | $\neg p$ | *alpha* 1 |
| | $X$ | 3 4 |

(iii)

| | | |
|---|---|---|
| 1 | $(p \to p) \to q$ | *premise* |
| 2 | $\neg q$ | *negated conclusion* |
| 3 | $\neg(p \to p)$ | *beta* 1 2 |
| 4 | $p$ | *alpha* 3 |
| 5 | $\neg p$ | *alpha* 3 |
| | $X$ | 4 5 |

Total [6]

(d)

| | | | |
|---|---|---|---|
| 1 | $\neg(p \wedge \neg q) \vee \neg(\neg s \wedge \neg t)$ | | *premise* |
| 2 | $\neg(t \vee q)$ | | *premise* |
| 3 | $u \to (\neg t \to (\neg s \wedge p))$ | | *premise* |
| 4 | $\neg\neg u$ | | *negated conclusion* |
| 5 | $u$ | | $\neg\neg$ 4 |
| 6 | $\neg t \to (\neg s \wedge p)$ | | *beta* 3 5 |
| 7 | $\neg t$ | | *alpha* 2 |
| 8 | $\neg q$ | | *alpha* 2 |
| 9 | $\neg s \wedge p$ | | *beta* 6 7 |
| 10 | $\neg s$ | | *alpha* 9 |
| 11 | $p$ | | *alpha* 9 |
| 12 | *branch* 1 | $p \wedge \neg q$ | *pb*1 |
| | 13 | $\neg(\neg s \wedge \neg t)$ | *beta* 1 12 |
| | 14 | $\neg\neg t$ | *beta* 13 10 |
| | 15 | $t$ | $\neg\neg$ 14 |
| | 16 | $X$ | 7 15 |
| 17 | *branch* 2 | $\neg(p \wedge \neg q)$ | *pb*2 |
| | 18 | $\neg\neg q$ | *beta* 17 11 |
| | 19 | $q$ | $\neg\neg$ 18 |
| | | $X$ | 8 19 |

Grand Total [20]

6
(a) [bookwork]
$wff$ ::= $\Box$ $wff$ | $\Diamond$ $wff$

Kripke model M
M = <W, R, ||>
Where W is non-empty set of worlds
R is accessibility relation on W
|| is denotation function which maps propositions onto subsets of W

Meaning of modal formulas

$\models_{M,a} \Box p$ *is true* $\leftrightarrow \forall w \,.\, aRw \rightarrow \models_{M,w} p$
$\models_{M,a} \Diamond p$ *is true* $\leftrightarrow \exists w \,.\, aRw \wedge \models_{M,w} p$

[6]

(b) [application]
*reflexive* $\forall w\ wRw$
*symmetric* $\forall ab\ aRb \rightarrow bRa$
*transitive* $\forall abc\ aRb \wedge bRc \rightarrow aRc$
*serial* $\forall w\ \exists x\ wRx$

[4]

(c) [application]
Pick B
(i)
M = <W,R,P>
W = {a,b}
R = {aRb}
||p|| = {a}
p is true in a
by MP box dia p true
so dia p true in b
but dia p false in b

assume p, show box dia p
suppose p is true in some a
suppose a R b, any b
then b R a by symmetry
so dia p at b
since dia p is true at any (every) b accessible from a
then box dia p is true at a, as required

[4]

(d)

| | |
|---|---|
| 1 | $\neg[\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)]$ |
| 1 | $\Box(p \rightarrow q)$ |
| 1 | $\neg(\Box p \rightarrow \Box q)$ |
| 1 | $\Box p$ |
| 1 | $\neg\Box q$ |
| 2 | $\neg q$ |
| 2 | $p$ |
| 2 | $p \rightarrow q$ |
| 2 | $q$ |
| | $X$ |

| | |
|---|---|
| 1 | $\neg[\,\Box p \to p\,]$ |
| 1 | $\Box p$ |
| 1 | $\neg p$ |
| 1 | $p$ |
| | $X$ |

| | |
|---|---|
| 1 | $\neg[\,p \to \Box\Diamond p\,]$ |
| 1 | $p$ |
| 1 | $\neg\Box\Diamond p$ |
| 2 | $\neg\Diamond p$ |
| 1 | $\neg p$ |
| | $X$ |

| | |
|---|---|
| 1 | $\neg[\,\Box p \to \Box\Box p\,]$ |
| 1 | $\Box p$ |
| 1 | $\neg\Box\Box p$ |
| 2 | $\neg\Box p$ |
| 3 | $\neg p$ |
| 3 | $p$ |
| | $X$ |

| | |
|---|---|
| 1 | $\neg[\,\Box p \to \Diamond p\,]$ |
| 1 | $\Box p$ |
| 1 | $\neg\Diamond p$ |
| 1 | $p$ |
| 1 | $\neg p$ |
| | $X$ |

| | |
|---|---|
| 1 | $\neg[\,\Diamond p \to \Box\Diamond p\,]$ |
| 1 | $\Diamond p$ |
| 1 | $\neg\Box\Diamond p$ |
| 2 | $p$ |
| 3 | $\neg\Diamond p$ |
| 2 | $\neg p$ |
| | $X$ |

[6]

Grand Total [20]