E3.16
ISE3.23

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2009

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

Corrected Copy

None

ARTIFICIAL INTELLIGENCE

Tuesday, 5 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) (i) Briefly describe uniform-cost and best-first graph search strategies.

(ii) Give a node representation for use with the General Graph Search (GGS) algorithm, which could be used for implementing both uniform-cost and best-first graph search strategies.

(iii) Briefly explain how the two different strategies can be implemented by changes to the `choose/3` and `add_to_paths/3` relations of the GGS algorithm.

(iv) Using appropriate criteria for evaluating graph search algorithms, compare and contrast uniform cost and best-first search.

[8]

b) You are presented with a simple propositional planning problem and you suspect that you might be able to solve it with the General Graph Search (GGS) algorithm.

There are three actions, I, J, and K. Action I has precondition (constraint) $\{\neg X\}$ and effects $\{X, Z\}$, action J has preconditions $\{\neg Y\}$ and effects $\{X, Y\}$, and action K has preconditions $\{X, Y, Z\}$ and effect $\{W\}$.

The initial state is $\{\neg W, \neg X, \neg Y, \neg Z\}$ and the goal state is $\{W\}$.

Formulate, in Prolog or other declarative notation, this planning problem, such that it could be used with the General Graph Search (GGS) algorithm.

(i) Specify a representation for the state.

(ii) Specify an initial state and a goal state.

(iii) Specify the state change rules.

(iv) Draw the search space (graph) and show the solution path.

[8]

c) Is the state-change $\{\neg W, \neg X, \neg Y, \neg Z\} \rightarrow I \rightarrow \{W, X, \neg Y, Z\}$ valid?

What would be the issue in trying to formalise this problem directly in propositional logic, and using a suitable formalism for representing time and action to reason about the solution to the problem?

[4]

2 a) Briefly describe the A* search algorithm, indicating what the *f*-cost of a node represents, which node on the search frontier will be expanded first, and which nodes in the search space will be expanded using A* search.

[4]

b) In A* search, explain why heuristic cost function h must be *admissible*. Use this property to explain why A* search is optimal.

[6]

c) A graph G is (explicitly) defined by a 3-tuple $G = \langle N,E,R \rangle$, where N is a set of nodes, E is a set of edges, and R is the incidence relation. A rooted graph has a unique node $s \in N$ from which all paths originate.

Give an inductive definition of the paths in a rooted graph G.

Give an alternative definition of a rooted graph. Show that the paths in the alternative definition are the same as the original (explicit) definition, clearly stating any assumptions you make.

Explain how A* search explores the paths induced by the alternative defintion.

[6]

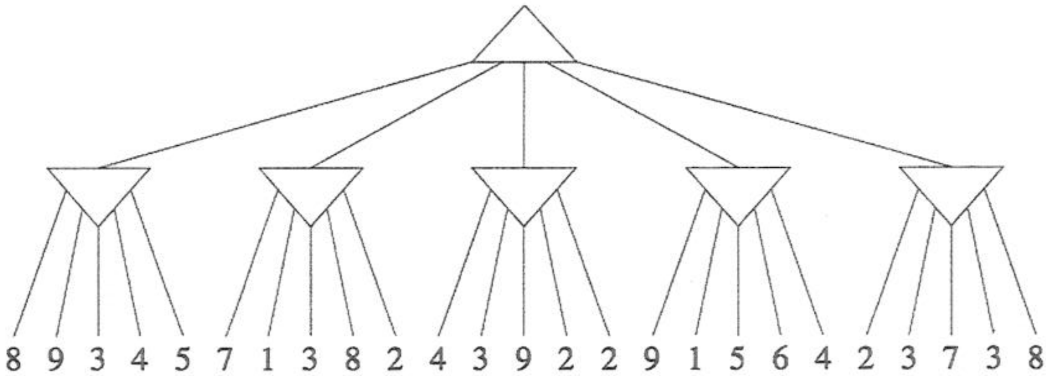
d) Suppose there are two heuristics, *happy* and *gloomy*, that can be applied to a node in the search space, and that for all nodes n, $gloomy(n) \leq happy(n)$.

Explain why *happy* may perform "better" (be more efficient in exploring the search space) than *gloomy*, even though the space complexity of A* search is still exponential with both heuristics.

[4]

3 a) Describe the Minimax Algorithm for two-player games.

Illustrate your answer using the following game tree, assuming it is MAX to move first.



[5]

b) Explain the limitations of Minimax search, and suggest some modifications to make it more generally applicable.

[4]

c) Describe the alpha-beta pruning search algorithm. Use the same game tree as above to illustrate your answer, in particular showing which branches are pruned.

[7]

d) Suggest how the ordering of nodes can improve the 'efficiency' of the search.

Suggest how a game playing program can be improved by using several heuristics.

[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) Define *unification*, and give a unification algorithm for two terms. If there is one, give the (most general) unifier of:

(i)	$p(X, q(Y), r(Z))$	and	$p(r(a), q(X), r(b))$
(ii)	$knows(John, X)$	and	$knows(Y, mother(Y))$
(iii)	$f(X)$	and	$f(g(X))$

[5]

c) Explain the *resolution rule* in propositional and in predicate logic.

Consider the following syllogism of logical inference:

All labour politicians are people who have two houses.
All people who have two houses are expense fiddlers.
Therefore all labour politicians are expense fiddlers.

Demonstrate that the syllogism follows from unification and resolution.

[5]

d) Consider the following formulas of First Order Predicate Logic

$\forall X.\forall Y.\ p1(X,Y) \rightarrow q(X,Y)$
$\forall X.\forall Y.\ p2(X,Y) \rightarrow r(X,Y)$
$\forall X.\forall Y.\ \forall Z.\ q(X,Y) \wedge r(X,Z) \rightarrow s(X)$
$p1(harry, ralph)$
$p2(harry, greg)$

Convert these formulas into Horn Clauses.

Prove, using resolution and showing the unifiers, that $s(harry)$.

[6]

5 a) Explain what is meant by the following statements, concerning a set of formulas S, a proposition p, and the calculus KE:

(i) $S \models p$

(ii) $S \vdash_{KE} p$

(iii) KE is sound,

(iv) KE is complete.

[4]

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

[4]

c) Consider the following sentences.

If Ken passes the AI exam, then Jem passes the HCI exam.
If Jem passes the HCI exam, then Ken passes the SE exam.
Ken passes either the AI exam, or the SE exam, but not both.

Formalise these statements in propositional logic.

Show, using the calculus KE, that *Ken passes the SE exam.*

Using the KE calculus, or otherwise, show whether (or not) *Jem passes the HCI exam.*

[6]

d) As well as using KE for proof by refutation, KE can be used for model building, i.e. apply the KE rules and look for (ideally) one open branch.

Now consider the island of knights and knaves. Everyone is either a knight, or a knave, but not both. Knights always tell the truth, and knaves always lie.

You meet two islanders, *Tony* and *Gordon*. They say to you:

Tony: I am a knave if (and only if) Gordon is a knave.
Gordon: We are of different kinds.

Who is which?

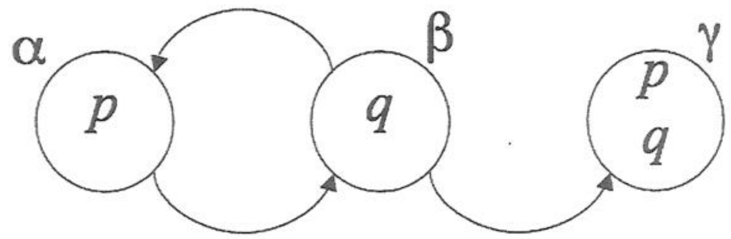
[6]

6 a) Describe how the syntax of propositional logic can be extended to give modal logic.

[2]

b) Define a *Kripke model*, and specify how a Kripke model is used to give a semantics for formulas of modal logic.

Consider the following diagram of a set of possible worlds.



Give the Kripke model represented by this diagram.

[6]

c) Given the Kripke model and semantics of part (b), say, with justification, whether each of the following formulas of modal logic is true or false:

(i) $M,\alpha \models \Diamond(\neg\Box p \vee \Diamond\neg q)$

(ii) $M,\gamma \models \Box p$

(iii) $M,\gamma \models \Diamond q$

(iv) $M,\gamma \models \Box(p \wedge q) \rightarrow \Diamond(p \wedge q)$

[4]

d) Consider the axiom schema **D** (seriality: $\forall a.\exists b.aRb$). Show that this axiom schema does not hold in the class of all models. Show that the axiom schema **D** does hold in the class of all models which are serial.

[4]

e) Using the KE tableau procedure for modal logic, show that:

$$(\Box p \rightarrow \Diamond p) \leftrightarrow (\neg\Box p \vee \neg\Box\neg p)$$

is a theorem of S5. Annotate your proof to show which rules have been used.

[4]

1 (a) bookwork

(i) uniform cost, expand node on search frontier with least cumulative path cost from the start node as computed by cost function given by *g*

best first, expand node on search frontier with least estimated path cost to a (the) goal node a computed by heuristic function given by *h*

[2]

(ii) node representation (state, g-cost, h-cost)

state: the actual representation of the problem
g-cost: actual path cost from start state to node
h-cost: estimated path cost from node to goal state

[2]

(iii) graph representation is a list of paths. So, either

choose: just picks first path in order, IF
add_to_paths: sort in ascending g-cost (resp. h-cost) for uniform (resp. best f)

OR

choose: traverses list of paths to find the cheapest, IF
add_to_paths: just appends new paths to old paths irrespective of cost

[2]

(iv) uniform cost:

complete: yes
optimal: yes provided path cost function is monotonic
time/space complexity: O(b^d)

best first:

complete: no
optimal: no
time/space complexity: O(b^m) but worst case reduced by good heuristic

[2]
Total [8]

(b) application

(i) representation: 4-tuple each representing one propositional variable, 0 for –P and 1 for P

[1]

(ii) initial state (0,0,0,0)
goal state (1,_,_,_)

[2]

(iii)

```
state_change( I, (W, 0 Y, _), (W, 1, Y, 1) ).

state_change( J, (W, _, 0, Z), (W, 1, 1, Z) ).

state_change( K, (_, 1, 1, 1), (1, 1, 1, 1) ).
```

[3]

(iv)
0000 -> I -> 0101 -> J -> 0111 -> K -> 1111
|-> J -> 0110

[2]
Total [8]

c)
Not in this formulation, because we have explicitly made sure that anything in the old state is unaffected by an action, stays the same in the new state.

[1]

The issue in a logical representation is the frame problem: there is nothing to say that the I action does not change the value of W. So we could have models which after the I action where W was true and would still be valid, unless we had frame axioms which said that anything true before the action and unchanged by the action stays true after the action.

[3]
Total [4]

Grand total [20]

2

(a) [bookwork]

A* search algorithm adds, for each node, g-cost + h-cost to give f-cost,

[1]

where the f-cost of node n represents the least estimated path cost from start state to a goal state through node n

[1]

so that A* chooses to expand node on search frontier with lowest f-cost

[1]

so that, given f* as the actual cost of getting to goal node G (h(G) = 0, so g(G) = f*):

- A* expands all nodes s.t. f(n) < f*
- A* may expand some nodes for which f(n) = f* (including goal node G)
- A* will not any expand any node for which f(n) > f*

[1]

Total [4]

(b) [bookwork/application]

Admissible => never over-estimate

If overestimate might expand sub-optimal goal G'

[2]

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

f(G')	=	g(G') + h(G')	
	=	g(G') + 0	G' is a goal state, we require h to be 0
	=	g(G')	

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 ^{3}
	f*	$\geq$	g(G')	a contradiction

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

[4]

Total [6]

(c) [bookwork/application]
We represent a path by a tuple (path,f-cost)

Where f-cost is given by edge (e) [ie the g-cost] plus h(n)

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

Pi+1 = {(p ++ <n>, (g+e+h(n)) | ∃(p,g) ∈ Pi. frontier(p) = (n') & (n',e,n) ∈ R }

[1]

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

[1]

Then the inductive definition of the paths graph is given by:

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

[1]

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

[1]

Assumption that Op computes every element of the incidence relation.

[1]

Let f* be the actual cost of getting from the start state to the optimal goal state. Then the A* algorithm constructs:

All the paths (p,f) in whichever P'_i such that f < f*
Some of the paths (p,f) in whichever P'_i such that f = f*
None of the paths (p,f) in any P'_i such that f > f*
At each step, A* selects the path in (p,f) from whichever P'_i such that this f is least so far, and expands that.
Clearly we construct a path in some Pi in which i=d (depth of solution), and might not construct the paths in some Pi i<d at all.

[1]
Total [6]

(d)
Let f* be cost of optimal node.
A* expands all nodes with f-cost less than f*.
A* expands some nodes with f-cost = f*.
A* expands no nodes with f-cost > f*.
Since f(n) = g(n) + h(n), this means that A* expands all those nodes such that
h(n) < f* - g(n).
In other words, the more nodes for which this relation holds, the more nodes will be expanded by A* using this heuristic, and the less efficiently will the search space be explored.
Alternatively, consider histogram of nodes according to actual f-cost, whereby
f-actual(n) = g(n) + h-actual(n).

```
|           |     *   *
|   *       * *   *   *
| * *   *   G * * *   *
| * * * * * * * * * * *
________________________
           f*
```

Only those nodes to the left of the f* bar will be expanded. In practice of course, we don't have h-actual we just have h. Thus we could have a redistribution of the histogram which pushes more of the nodes to the left of the f* bar.
In other words, we want to ensure f* < f(n) + h(n) < f-actual(n)

[3]

Happy performs better than gloomy because the exponential is worst case complexity. "On average" we expect to do better than this.

[1]
Total [4]

Grand Total [20]

3

(a) [bookwork/application]

max is player trying to win, or MAXimize advantage

min is opponent who attempts to MINimize max's score. Assume that min uses the same information as max and attempts always to move to a state that is worst for max

each leaf node is given a score of 1 or 0, depending on whether the state is a win for max or min respectively

exhaustively generate the graph

propagate leaf values up the graph according to the rules:

 if the parent is a MAX, give it the minimum value of its children

 if the parent is a MIN, give it the maximum value of its children

[2]

Game Tree

MIN nodes are labelled (l to r): 3 1 2 1 2

MAX node (root) labelled 3

[3]

Total [5]

(b)[bookwork/application]

exhaustive search is not always possible because of exponential space

[2]

search to fixed ply

apply heuristic to leaf nodes

propagate as before

[2]

Total [4]

(c))[bookwork/application]

➠Associate one of two values with each node

—Alpha value, associated with MAX nodes, which can never decrease

•Alpha is the 'least' MAX can get, given MIN will do its best to minimise MAX's value

-- Beta value, associated with MIN nodes, which can never increase

• Beta is the 'most' MAX can get, given MIN will do its best to minimise MAX's value

➢ **Algorithm**

➠Search to full ply using depth first

➠Apply heuristic evaluation to all siblings at ply

—Assume these are MIN nodes

➠Propagate value of siblings to parent using Minimax rules

—If MIN nodes, back up the maximum value

➠Offer this value to ***grandparent*** MIN node as possible beta cutoff

➠Descend to other grandchilren

➠Terminate (prune) exploration of parent if any of their values is greater than or equal to the beta cutoff

➠Do the same for MAX nodes

➠Two rules for terminating search

–Search stopped below any MIN node having a beta value
less than or equal to alpha value of any of its MAX ancestors
Search stopped below any MAX node having an alpha value
greater than or equal to beta value of any of its MIN ancestors

[3]

Game Tree
First branch not pruned at all; 3 is offered as beta cut-off to all child MIN nodes

Second branch pruned after the 1 returned
(any value greater than 1 will 'lose' to 1 when determining the beta value of the MIN node)
(since 1 < 3, any value less than 1 will lose to the 3 when determining the alpha value of the MAX node)
(why you don't prune after the 7 returned
suppose every other value was 6, we want to return 6, not 7 or 3)

Third branch pruned after the 2 (fourth sub-branch)

Fourth branch pruned after the 1

Fifth branch pruned after 2

[4]
Total [7]

(d) [application/understanding]
Ideally want the MIN children of a MAX node in descending order.
Ideally want the MAX children of a MIN node in ascending order.
Then the cutoff has most chance of 'kicking in' quickest.

[1]

Use weighted sum of multiple heuristics

Weight indicates relative importance.

Adjust weights after each game according to result => learning.

[3]
Total [4]

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/application]
unification
unification of terms s and t is u (if it exists) which is a term that is a substitution instance of both s and t (ie a term that results from a consistent substitution of constants for variables)

[1]

algorithm
two constant unify if the are the same constant
a variable unifies with a term
two compound terms unify if
    they have the same functor
    they have the same arity (no arguments)
    the arguments piecewise unify

[1]

(i) { X/r(a), Y/X, Z/b }
(ii) { Y/John, X/mother(Y) }
(iii) { f(f(… f(X) …)) [hope you've got the occurs check on :-]

[3]
Total [5]

(c) [bookwork/application]
resolution
inference rule which produces new clause from two other clause containing complementary literals

e.g. in propositional logic
$p \vee q$
$\neg q \vee r$
‾‾‾‾‾‾
$p \vee r$

[2]

Let
p = *labour politician*
q = *have two houses*
r = *expense fiddlers*

$\forall x.p(x) \rightarrow q(x)$
$\forall y.q(y) \rightarrow r(y)$

is equivalent to (dropping universal quantifier

$\neg p(x) \rightarrow q(y)$
$\neg q(y) \rightarrow r(y)$

$p(x) \vee r(y) \qquad \{x/y\}$

[3]
Total [5]

(*d*) [*application*]

$\forall X. \forall Y.\ p(X,Y) \rightarrow q(X,Y)$
$\forall X. \forall Y.\ p(X,Y) \rightarrow r(X,Y)$
$\forall X. \forall Y.\ \forall Z.\ q(X,Y) \wedge r(X,Z) \rightarrow s(\ X\ )$
$p(\ harry, ralph\ )$
$p(\ harry, greg\ )$

$f1 \quad \neg p1(X1, Y1) \vee q(X1, Y1)$
$f2 \quad \neg p2(X2, Y2) \vee r(X2, Y2)$
$f3 \quad \neg q(X3, Y3) \vee \neg r(X3,Z) \vee s(\ X3\ )$
$f4 \quad p1(\ harry, ralph\ )$
$f5 \quad p2(\ harry, greg\ )$

[2]

$\neg s(harry)$

[1]

resolve with $f3$, *unifier* $\{X3\ /\ harry\}$
$\neg q(harry, Y3) \vee \neg r(harry,Z)$
resolve with $f1$, *unifier* $\{X1\ /\ harry,\ Y1/\ Y3\}$
$\neg p(harry, Y1) \vee \neg r(harry,Z)$
resolve with $f4$, *unifier* $\{\ Y1\ /\ ralph\}$
$\neg r(harry, Z\}$
resolve with $f2$, *unifier* $\{X2\ /\ harry,\ Z\ /\ Y2\ \}$
$\neg p(harry, Y2)$
resolve with $f5$, *unifier* $\{Y2\ /\ greg\ \}$
[*empty clause*]

[3]
Total [6]

Grand Total [20]

5

(a) [bookwork]

(i) S entails p: any valuation (assignment of truth values to propositional symbols) which makes every formula in S true, also makes p true

[1]

(ii) S proves p: 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 |-KE (S -> p) and we use refutation, but the |- relation is the same)

[1]

(iii) sound: if KE proves something, it is an entailment, i.e. |- implies |=

[1]

(iv) complete: if there is an entailment, then KE can prove it, i.e. |= implies |-

[1]

Total [4]

(b) [bookwork/understanding]

- *KE proof procedure*
  - *to prove Q, begin with ¬Q and search for a refutation*
  - *expand ¬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*

Total [4]

c) [application]

Let the symbols

KpassAI
KpassSE
JpassHCI

Have the obvious intuitive representation

Then

pr1	$KpassAI \rightarrow JpassHCI$
*pr*2	$JpassHCI \rightarrow KpassSE$
*pr*3	$KpassAI \vee KpassSE$
*pr*4	$\neg(KpassAI \wedge KpassSE)$

[2]

Try to prove *KpassSE* so negate conclusion and proceed:

1	$\neg KpassSE$	
2	*KpassAI*	*beta*, 1, *pr*3
3	*JpassHCI*	*beta*, 2, *pr*1
4	*KpassSE*	*beta*, 2, *p*2
	close	1,4

[2]

JPassHCI
Intuitively, if KpassSE then ¬KpassAI, and pr3 and pr4 are both true.
But then either JpassHCI OR JpassHCI will make pr1 and pr2 true.

Intuitively, if KpassAI then the only way to make p1 true is for JpassHCI to be true.
Then the only way to make pr2 true is to make KpassSE true.
But then this contradicts pr4.

Note that trying to prove KpassAI (so add ¬KpassAI) leaves an open branch

> *¬KpassAI*
> *KpassSE*
> *Open (pr3 analysed, pr1, p2 and pr4 analysed by beta simplification)*

Trying to prove ¬KpassAI (so add KpassAI) closes:

> *¬KpassSE*
> *JpassHCI*
> *¬KpassAI*
> *close*

Trying to prove either *JpassHCI* or *¬JpassHCI* both do not close. This leads on to…

[2]
Total [6]

d) [application/understanding]

First off note we have:

knave(tony) → ¬knight(tony)	*knight(tony) → ¬knave(tony)*
knave(gordon) → ¬knight(gordon)	*knight(gordon) → ¬knave(gordon)*

[1]

Now, the truth or falsity of what they say depends on what kind they are, i.e.:

knave(tony) → ¬(knave(tony) ↔ knave(gordon))
knight(tony) → (knave(tony) ↔ knave(gordon))

knave(gordon) →
> *¬ [(knave(tony) ∧ knight(gordon)) ∨ (knight(tony) ∧ knave(gordon))]*

knight(gordon) →
> *[(knave(tony) ∧ knight(gordon)) ∨ (knight(tony) ∧ knave(gordon))]*

[2]

OK: Let's build a model.

Add that gordon is a knight:

knight(gordon)
[(knave(tony) ∧ knight(gordon)) ∨ (knight(tony) ∧ knave(gordon))]

branch 1
(knave(tony) ∧ knight(gordon))
knave(tony)
knight(gordon)
¬(knave(tony) ↔ knave(gordon))
¬knave(gordon)
knight(gordon)
open branch:

branch 2
$\neg(knave(tony) \wedge knight(gordon))$
$(knight(tony) \wedge knave(gordon))$
$knight(tony)$
$knave(gordon)$
$\neg knight(gordon)$
close

[3]

So you there have the model: Tony is a knave and Gordon is a knight.

Actually, we all knew Tony is a liar. [*So is Gordon actually but that is another story.*]

Total [6]

Grand Total [20]

6

(a)

wff ::= $\Box$ wff | $\Diamond$ wff

Total [2]

(b) [application]

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

[3]

W = {α,β,γ,}

R = { α R β, β R a, β R g, }

|p| = { α, γ }

|q| = { β, γ,}

[3]

Total [6]

(c) [application]

(i) true

only worlds accessible from a is b

so evaluate $\neg\Box p \vee \Diamond\neg q$ at b

first disjunct is false because for worlds accessible from b, p is true

but second disjunct is true, since there is a wrld accessible (a) where q is not true

[1]

(ii) true

box p is always true at the end of a chain because the semantics has ∀ … → ...

the antecedent is always false therefore the forall is true…

[1]

(iii) false

dia q is never true at the end of a chain, since there is no world accessible

[1]

(iv) false

this is of course the seriality axiom schema dressed up, and it is not

[1]

Total [4]

(d) [application]

One way is to point to part iv above

Or let M = <W,R,P>

W = {a }

R = {}

||p|| = {a}

box p is true in a

dia p is false in a

but if D holds then dia p true in a

this is a contradiction

[2]

assume |=M,a box p, show |=M,a dia p
suppose there is some (any) a in some (any) M
by seriality there is some b s.t aRb
if |=M,a box p then |= M, b p
so |=M,a dia p as required

[2]
Total [4]

(e) [application]

$$(\Box p \rightarrow \Diamond p) \leftrightarrow (\neg\Box p \vee \neg\Box\neg p)$$

proof by refutation, so

$$\neg((\Box p \rightarrow \Diamond p) \leftrightarrow (\neg\Box p \vee \neg\Box\neg p))$$

branch 1

$(\Box p \rightarrow \Diamond p)$
$\neg(\neg\Box p \vee \neg\Box\neg p)$
$\neg\neg\Box p$
$\neg\neg\Box\neg p$
$\Box p$
$\Box\neg p$
p
$\neg p$
close

[2]

branch 2

$\neg (\Box p \rightarrow \Diamond p)$
$(\neg\Box p \vee \neg\Box\neg p)$
$\Box p$
$\neg\Diamond p$
p
$\neg p$
close

[2]
Total [4]

Grand Total [4]