EE3-16

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2013

EEE/EIE PART III/IV: MEng, Beng and ACGI

Corrected Copy

ARTIFICIAL INTELLIGENCE

Thursday, 10 January 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) : T-K. Kim

The Questions

1 a) Specify the General Graph Search (GGS) algorithm and explain how it is used in problem solving search to find a solution path in a graph.

[10]

b) Explain how a search space needs to formulated (specified) so that it can be explored using the GGS algorithm.

[4]

c) There is a cat, which can occupy one of four locations: hall, kitchen, garden, or living room. The kitchen is accessible from the hall (and vice-versa) by a door if the door is open; the living room is accessible from the hall (and vice-versa) by another door if that door is open; the garden is accessible from the living room (and vice versa) by a window if the window is open; and the kitchen is accessible from the garden (and vice versa) by a cat-flap, provided the cat-flap is unlocked.

There is a bowl of food in the kitchen. The cat is in the hall. The cat wants to get to the kitchen to eat the food. The kitchen-hall door is closed, the hall-living room door is open, the window is open, and the cat-flap is unlocked.

The cat can perform the following actions: it can open/close or lock/unlock a door, window or cat-flap by staring at it; or it can move from one location to another provided it is accessible.

Formulate (specify in Prolog or other declarative notation) a search space for the problem, so that the cat could solve it with the GGS algorithm.

Assuming the cat is implemented in Prolog and uses breadth-first search, give the solution path to the problem from the specified state.

[6]

2 a) Explain how Uniform Cost, Best First, and the A* graph search algorithms work, and compare and contrast the performance of the three algorithms with respect to appropriate criteria.

[6]

b) Consider the graph G shown in Figure 1.

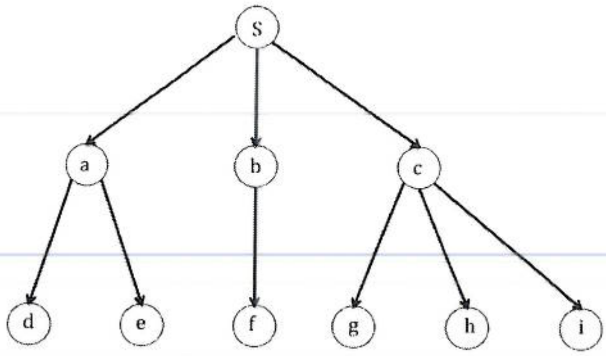


Figure 2.1: Graph G

i) Give a formal (explicit) definition of the graph G.

ii) From start node S, give an inductive definition of the paths P_G in the graph G, and say what they are.

iii) Specifying any partial functions that might be required, give an implicit definition of the graph G'.

iv) Use the definition of Part (iii) to give another inductive definition of the paths P_G' defined by graph G'.

v) State what requirement must be satisfied for $P_G = P_G'$.

[10]

c) Given a goal state, explain how the Uniform Cost search algorithm explores the paths defined by G'.

[4]

3 The game rock-paper-scissors-lizard-Spock is a 2-player game, played as follows.

The first player makes a gesture representing one of rock, paper, scissors, lizard or Spock.

The second player makes a gesture representing one of rock, paper, scissors, lizard or Spock.

The winner is decided by the following set of rules:

- Scissors cuts paper.
- Paper covers rock.
- Rock crushes lizard.
- Lizard poisons Spock.
- Spock smashes scissors.
- Scissors decapitates lizard.
- Lizard eats paper.
- Paper disproves Spock.
- Spock vaporizes rock.
- Rock crushes scissors.

a) Formulate the problem (specify in Prolog or other declarative notation) as a search space.

[8]

b) Describes the minimax algorithm, and briefly explain how the General Graph Search (GGS) algorithm could be adapted to implement the minimax algorithm.

[8]

c) This game is actually played with each player selecting the gesture concurrently, and with several rounds (best of three, best of five, etc.)

What are the limitations of the minimax for playing this sort of game, in this sort of situation? Give some indication of how you might use the alpha-beta algorithm for deciding what gesture to select in any round.

[4]

5 a) Explain how the KE proof procedure builds a tableau (KE-tree) to show that a set of propositional statements comprising a knowledge base KB entails a single proposition p (i.e. that $KB \models p$).

[8]

b) Prove, using the proof procedure KE, the following formulas:

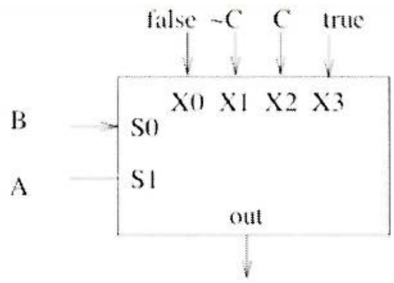
i) $((p \wedge q) \rightarrow r) \leftrightarrow (p \rightarrow (q \rightarrow r))$

ii) $p \rightarrow ((p \wedge q) \vee (p \wedge \neg q))$

[6]

c) A multiplexer can be used to implement random logic. A 2^{n+1} multiplexer can implement any random logic function of $n+1$ variables.

Consider the multiplexer shown in Figure 5.1.



A	B	C	out	
F	F	F	F	*false*
F	F	T	F	
F	T	F	T	~C
F	T	T	F	
T	F	F	F	C
T	F	T	T	
T	T	F	T	*true*
T	T	T	T	

Figure 5.1: Multiplexer implementation of random logic function

Prove, using the KE proof procedure, that $out \equiv ((B \wedge \neg C) \vee (A \wedge C))$.

It is essential that you annotate the KE-tree to show the inference steps.

[6]

6 a) Define a syntax for well-formed formulas of propositional modal logic. Explain how a Kripke model is used to give a semantics for well-formed formulas of propositional modal logic.

[4]

b) Consider the Kripke model $M = \langle\{\alpha,\beta\},\{\alpha\alpha,\alpha\beta\},||\rangle$, where $|p| = \{\beta\}$.

Explain whether the following formulas are true or false.

i) In world α, $\Box p$

ii) In world α, $\Diamond p$

iii) In world β, $\Box p$

iv) In world β, $\Diamond p$

[4]

c) Consider the following axiom schemas. State the corresponding frame condition (if any) on the class of all models.

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]

d) Prove, using the KE calculus for propositional modal logic, that all six axiom schemas in Part (c) hold in the modal logic S5.

[6]

The Answers

1.

(a)

state is a representation of the state of the problem space
node is collection of information, including state.
path is a list of nodes
graph is a list of paths

```
search( Paths, [Node|Path] ) :-
        choose( [Node|Path], Paths, _ ),
        state_of( Node, State ),
        goal_state( State ).
search( Paths, SolnPath ) :-
        choose( Path, Paths, OtherPaths ),
        one_step_extensions( Path, NewPaths ),
        add_to_paths( NewPaths, OtherPaths, AllPaths ),
        search( AllPaths, SolnPath ).
```

Search graph for a solution path, if
    Pick a path, ignore rest
    Get the state of the frontier node
    Is it a goal state – yes: path is a solution path
Search graph for a solution path, if
    Pick a path, rest is set of OtherPaths
    Get the state of the frontier node
    Apply all the operators to the state
    Add all the nodes to the front of path, give set of NewPaths
    Add NewPaths to OtherPaths to make a BiggerGraph
    Search BiggerGraph for a solution path

(b)

state representation

state transformers

initial state

start state

(c)

state representation
(Location of Cat, Location of Food, (Door1, Door2, Window, CatFlap))

```
state_change( stare, (hall, Food, (S1,S2,S3,S4)), (hall, Food, (S1new,S2,S3,S4)) ) :-
        opposite( S1, S1new ).
Etc.
state_change( move, (hall, Food, (S1,S2,S3,S4)), (kitchen, Food, (S1,S2,S3,S4)) ) :-
        accessible( hall, kitchen, S1 ).
Etc.
```

Opposite(open, closed).
Etc.

Accessible(hall, kitchen, open).

Initial state
(hall, kitchen (closed, open, open, unlocked))

Goal state
(kitchen, kitchen, _)

(hall, kitchen (closed, open, open, unlocked)) → stare →
(hall, kitchen (open, open, open, unlocked)) → move →
(kitchen, kitchen, _)

So the cat keeps staring at the door until it forces it to open with the sheer power of its mind rather than searching for an alternative solution path

2
(a)
UC: pick path with frontier node with lowest actual cost given by path cost function g
BF: pick path with frontier node with lowest estimated cost given by heuristic h
A*: pick path with frontier node with lowest estimated cost of path through node to goal given by g + h

UC: time space exponential, optimal complete
BF: time space exponential but reduced substantially with good heuristic, not optimal not complete
A*: still looking at exponential complexity but reduced substantially with good heuristic and optimal and complete

(b)
(i)
G = <N,R>
N = {S, a, b, c, d, e, f, g, h, i}
R = { Sa, Sb, Sc, ad, ae, bf, cg, ch, ci}

(ii)
PG = UPi

P0 = <S>

Pi+1 = { p ++ <n> | \exists p \in Pi . (frontier(p), n) \in R }

P1 = { <Sa>, <Sb>, <Sc> }

P2 = { <Sad>, <Sae>, <Sbf>, <Scg>, <Sch>, <Sci> }

P3 = P4 = … Pinfinity = \emptyset

(iii)
op1(S) = a, op2(S) = b, op3(S) = c
op1(a) = d, op2(a) = d
op1(b) = f
op1(c) = g, op2(c) = h, op3(c) = i

G' = < S, Op>

Op = {op1, op2, op2}

(iv)
P'0 = <S>

P'i+1 = { p ++ <n> | \exists p \in P'i . \exists op \in Op . op(frontier(p)) = n }

(v)
(a,b) \in R if-and-only-if \exists op \in Op . op(a) = b

(c)
Assume edges have cost associated with them

Start from P0

Compute all members of P1

Pick cheapest of P1 to compute some members of P2

Pick cheapest P1, partial P2, etc

3

(a)

state representation

(player-to-go, P1gesture, P2gesture)

```
state_change( p1move, (p1togo, P1g, P2g), (p2togo, P1g, P2g) ) :-
        gesture( P1g ).
state_change( p1move, (p2togo, P1g, P2g), (gameover, P1g, P2g) ) :-
        gesture( P2g ).

gesture( rock ).
gesture( paper ).
gesture( scissors ).
gesture( lizard ).
gesture( _Spock ).

Initial_state( p1togo, P1gesture, P2gesture).

Goal_state( gaemover, P1g, P2g ) :-
        cuts( P2g, P1g ).
Etc.

cuts( scissors, paper ).
Etc.
vaporises( _Spock, rock ).
```

(b)

Exhaustive search of tree
Assign leaf nodes
Propagate values up tree

Modify GGS to do exhaustive search

(c)

concurrent turns not turn taking
representation of other player not the same

use some machine learning to try to predict next move
use search to compute counter move (but this is overkill here)

4
(a)
soundness. if system proves something. it is an entailment

completeness. if something is an entailment. system can prove it

unification. algorithm that check if two terms can be unified (substitution of values for variables that makes the two terms syntactically equivalent)

resolution rule: p or q. -p or r. infer q or r

(b)
bookwork

(c)
\forall x \forall y . psych(x) & says_nutter(x,y) -> - soundmind(y)
\forall x . soundmind(x) -> responsible(x)
\forall x . - soundmind(x) -> - responsible(x)
\forall x . arsonist(x) & responsible(x) -> guilty(x)
\forall x . arsonist(x) & - responsible(x) -> guiltybutinsane(x)

-psych(x1) or -says_nutter(x1,y) or - soundmind(y)
-soundmind(x2) or responsible(x2)
soundmind(x3) or - responsible(x3)
-arsonist(x4) or -responsible(x4) or guilty(x4)
-arsonist(x4) or responsible(x4) or guiltybutinsane(x4)

(d)
psych(sigmund)
arsonist(X)
says_nutter(sigmund, X)

(e)
- guiltybutinsane(X)

-arsonist(X) or responsible(X) {x4=X}

responsible(X) {X=X}

soundmind(X) {x3=X}

-psych(x1) or -says_nutter(x1,X) {y=X}

-says_nutter(sigmund,X) {x1=sigmund}

contradiction

5
(a)
entailment KB = p
show KB -ke p
let KB' be distributed and over KB
deduction theorem - (KB' -> p)
prove by refutation -(KB' -> p)
root of Ke-tree is each premise in KB', negated conclusion -p by ->-elim and &-elim rules of KE
then proof procedure is for tableau T pick a branch \theta analyse with KE rules (in context if necessary) extend the the brachs with formulas form KE rules to produce tableau T'
repeat until every branch is closed (formual and it negation on every branch)

(b)
(i)
-[((p & q) > r) <> (p > (q > r))
branch 1
-((p & q) > r)
(p > (q > r))
p & q
-r
p
q
q > r
r
close

branch 2
((p & q) > r)
-(p > (q > r))
p
-(q > r)
q
-r

branch 2.1
p & q
r
close

branch 2.2
-(p and q)
-q
close

(ii)
-[p -> ((p & q) · (p & -q))]
p
-((p & q) · (p & -q))

-(p & q)
-(p & -q)
q
-q
close

(iii)
-A & -B > -out
-A & B > -C
A & -B > C
A & B > out
-[((B & -C) · (A & C)) <> out]

Branch 1
((B & -C) · (A & C))
-out
-A & -B
-A
-B

Branch 1.1
B & -C
B
-C
close

Branch 1.2
-(B & -C)
A & C
A
C
Close

Branch 2
-[((B & -C) + (A & C))]
out
-(B & -C)
-(A & C)
A & B
A
B
--C
C
-C
close

6

(a)

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* $\leftrightarrow \exists w\ .\ aRw \wedge\ \models_{M,w} p$

(b)

false, true, true, false

(c)

none

reflexivity

symmetry

transitivity

seriality

Euclidean

(d)

K

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	$p \rightarrow q$
2	q

close

reflexivity

1	$\neg(\Box p \rightarrow p)$
1	$\Box p$
1	$\neg p$
1	p

close

symmetry

1	$\neg(p \rightarrow \Box\Diamond p)$
1	p
1	$\neg\Box\Diamond p$
2	$\neg\Diamond p$
1	$\neg p$

close

transitivity

1	$\neg(\Box p \rightarrow \Box\Box p)$
1	$\Box p$

1 $\neg\Box\Box p$
2 $\neg\Box p$
3 $\neg p$
3 p
close

seriality
1 $\neg(\Box p \rightarrow \Diamond p)$
1 $\Box p$
1 $\neg\Diamond p$
1 $\neg p$
1 p
close

euclidean
1 $\neg(\Diamond p \rightarrow \Box\Diamond p)$
1 $\Diamond p$
1 $\neg\Box\Diamond p$
2 $\neg\Diamond p$
3 p
3 $\neg p$
close