EE3-16

1. a) [bookwork]

Depth-first: choose node on search frontier furthest from the start.

Depth-limited depth-first: stop depth-first if path depth d=limit.

Iterative deepening: depth-limited at first with limit=0, limit=1, limit=n. [3]

b) [bookwork]

Depth-first: optimal no, complete no, time complexity b^m (m=max depth of tree), space complexity $b \times m$.

Depth-limited: optimal no, complete no, time complexity b^l(l=limit), space complexity $b \times l$.

Iterative deepening: optimal yes, complete yes, time complexity b^l(l=limit), space complexity $b \times d$.

[3]

c) [application]

```
fib(0,0):- !.
fib(1,1):- !.
fib(N,F) :- N1 is N - 1 , N2 is N - 2, fib(N1,F1), fib(N2,F2), F is F1 + F2.

fibonacciN(X,[]) :- X <0, !.

fibonacciN(X,[Z|Z1]) :-
fib(X,Z),
X1 is X - 1,
fibonacciN(X1,Z1).
```

[4]

d) [application]

i) (D,JO,AN,JA) where represent the position of the driver, john, anna and james. The possible values are a, b or c.

ii) (a,a,a,a)

iii) (c,c,c,c)

iv)
```
statechange( driver, (D,JO,AN,JA), (O,JO,AN,JA) ) :-
distance(D,O,1),
\+ distance(O,JO,2),
\+ distance(O,AN,2),
\+ distance(O,JA,2),
safeB1(JO,AN),
safeB1(AN,JA),
safeB2(JO,JA),
safeB2(AN,JO).

statechange( john, (D,D,AN,JA), (O,O,AN,JA) ) :-
distance(D,O,1),
\+ distance(O,AN,2),
```

```
\+ distance(O,JA,2),
safeB1(AN,JA).

statechange( anna, (D,JO,D,JA), (O,JO,O,JA) ) :-
distance(D,O,1),
\+ distance(O,JO,2),
\+ distance(O,JA,2),
safeB2(JO,JA).

statechange( james, (D,JO,AN,D), (O,JO,AN,O) ) :-
distance(D,O,1),
\+ distance(O,JO,2),
\+ distance(O,AN,2),
safeB1(JO,AN),
safeB2(AN,JO).


distance(a, c, X) :- X = 2.
distance(c, a, X) :- X = 2.


distance(a, b, X) :- X = 1.
distance(b, a, X) :- X = 1.


distance(b, c, X) :- X = 1.
distance(c, b, X) :- X = 1.


safeB1(X,Y):- \+ X = Y.
safeB1(X,Y) :- X = Y, b2(X).


safeB2(X,Y):- \+ X = Y.
safeB2(X,Y) :- X = Y, b1(X).


b1(a).
b1(c).
b2(b).
```

One solution:
(anna,c,c,c,c)–(driver,b,c,b,c)–(james,c,c,b,c)–(driver,b,c,b,b)–(john,c,c,b,b)–(john,b,b,b,b)–(driver,a,a,b,b)–(james,b,a,b,b)–(driver,a,a,b,a)–(anna,b,a,b,a)–((is),a,a,a,a)

[10]

2. a) [bookwork]

Uniform-cost: choose node on search frontier with least actual cost from start node (cost function g).

Best-first: choose node on search frontier with least estimated cost to goal node (heuristic function h).

A*: choose node on search frontier with least estimated path cost from start node to goal node through n(f=g+h).

[4]

b) [bookwork]

Uniform-cost: optimal yes (successor of any node is equal to or greater than the cost of getting to that node), complete yes, time complexity b^d, space complexity b^d.

Best-first: optimal no, complete no, time complexity b^d, space complexity b^d. (worse case, does much better with good heuristic)

A*: optimal yes, complete yes, complexity depends on heuristic.

[4]

c) Questions:

i) Two variables for the location (x,y), one boolean for each type of alien.
(x $\in$[1:N], y $\in$[1:M], eatenA $\in$[T,F], eatenB $\in$[T,F], eatenC $\in$[T,F])

ii) $N \times M$ locations and 8 possible assignments to the boolean variables. Search space is $N \times M \times 8$.

iii) Each state has maximum four successors corresponding to the four possible actions. At most, the branching factor is 4.

iv) Initial State: (x,y,false,false,false)

v) Goal State: (_,_,true,true,true)

vi) Admissible heuristic example: smallest manhattan distance to any remaining alien of uneaten type.
Not admissible heuristic: number of aliens remaining. Not admissible because it overestimates the cost (increases the heuristic).

[12]

3. [application]

a) 4, can be determined by applying the AlphaBeta algorithm (or MiniMax)

[4]

b) True, The root node is a maximising node, the value of beta never changes at a maximising node.

[4]

c) Beta will take the smallest value returned by any of P's children. In this case, P's children return 7, 8 and 3, so the value of beta at A will be 3.

[4]

d) No.

[2]

e) Yes. At the leftmost subtree of C, returns the value of 2. Beta at C will then become 2, and 2 is less than the alpha value at C (4). So C will return the value immediately to its parent. The other two subtrees of C are pruned.

[2]

f) Having 4,8 and 9 and being a minimising node, will return 4 to its parent.

[2]

g) Since its beta value after visiting the leftmost subtree was less than the alpha value of 4, C will return its alpha value of 4.

[2]

4. a) [bookwork]
resolution: a rule of inference used for automated theorem proving.

rule:

$$\frac{\begin{array}{l} p \vee q \\ \neg p \vee r \end{array}}{q \vee r}$$

$$\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_{i-1} \vee (\neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m) \vee \neg p_{i+1} \vee \ldots \vee \neg p_m$$

$$\frac{\begin{array}{l} p \\ \neg p \end{array}}{\perp}$$

In logic programming: query and horn clauses. [4]

b) [bookwork]
unification: solving a problem of equating symbolic expressions.

algorithm:
uninstantiated variable unifies with atom, term, or other uninstantiated variable.
atom unifies with identical atom.
term unifies with term if functors same, arity same, pairwise arguments unify.

in logic programming: way of binding values to variables in resolution. [4]

c) [application]

$\forall x.flies(x) \wedge looks(x) \rightarrow superman(x)$
$\forall x.levitates(x) \rightarrow flies(x)$
$\forall x.cape(x) \rightarrow looks(x)$
$\forall x.cape(x) \wedge wearssimilar(x,y) \rightarrow cape(y)$
$\neg flies(X1) \vee \neg looks(X1) \vee superman(X1)$
$\neg levitates(X2) \vee flies(X2)$
$\neg cape(X3) \vee looks(X3)$
$\neg cape(X4) \vee \neg wearssame(X4, Y1) \vee cape(Y1)$

[4]

d) [application]

$cape(batman)$
$levitates(clarkkent)$
$wearssimilar(batman, clarkkent)$

[2]

e) [application]
Prove $superman(clarkkent)$.
Start with negated conclusion (proof by refutation)

$\neg superman(clarkkent)$
$\neg flies(clarkkent) \vee \neg looks(clarkkent) \{X1 = clarkkent\}$
$\neg levitates(clarkkent) \vee \neg looks(clarkkent) \{X1 = clarkkent, X2 = clarkkent\}$
$\neg looks(clarkkent) \{X1 = clarkkent, X2 = clarkkent\}$
$\neg cape(clarkkent) \{X1 = clarkkent, X2 = clarkkent, X3 = clarkkent\}$
$\neg cape(X4) \vee \neg wearssame(X4, clarkkent) \{X1 = clarkkent, X2 = clarkkent, X3 = clarkkent, Y1 = clarkkent\}$
$\neg cape(batman) \vee \neg wearssame(batman, clarkkent) \{X1 = clarkkent, X2 = clarkkent, X3 = clarkkent, Y1 = clarkkent, X4 = batman\}$
nil

[6]

5. a) [bookwork]

$wff ::= \Box wff \mid \Diamond wff$

Kripke model M:

$M = < W, R, || >$

where W is non-empty set of worlds

R is the accessibility relation on W

|| is the denotation function which maps propositions onto subsets of W

Meaning of modal formulas

$\models M, a\Box p$ is true $\leftrightarrow \forall w.aRw \rightarrow \models M, wp$

$\models M, a\Diamond p$ is true $\leftrightarrow \exists w.aRw \wedge \models M, wp$

[5]

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]

1	$1 : \neg(\Diamond\Box p \rightarrow \Box p)$	negated conclusion
2	$1 : \Diamond\Box p$	α, 1
3	$1 : \neg\Box p$	α, 1
4	$2 : \Box p$	poss, 2
5	$3 : \neg p$	poss, 3
6	$3 : p$	ness, 4
	$\times$	5, 6

1	$1 : \neg(\Diamond\Diamond p \rightarrow \Diamond p)$	negated conclusion
2	$1 : \Diamond\Diamond p$	α, 1
3	$1 : \neg\Diamond p$	α, 1
4	$2 : \Diamond p$	poss, 2
5	$3 : p$	poss, 4
6	$3 : \neg p$	ness, 3
	$\times$	5, 6

[4]

d) [bookwork]

Beliefs-Desires-Intentions (BDI) architecture with diagram (2 points)

- has its origins in the study of mental attitudes.
- beliefs: represent the agents informational state.
- desires: represent the agents motivational state.
- intentions: represent the agents deliberative state.

Beliefs (1 point)

- current knowledge about state of the world, some aspects of internal state.
- include facts about static properties of application domain.
- others acquired by agent as it executes plans.
- may need to represent meta-level beliefs and beliefs of other agents.

Desires/goals (1 point)

- conditions over some interval of time (or sequence of world states)
- can then have goals to achieve, test, maintain and wait for a condition

Plans (1 point)

- how to act when certain facts added to belief db, or new goals acquired.
- consist of: invocation, context and maintenance conditions, and a body.
- used to create instances of the plan to be executed.

Intentions (1 point)

- intention structure contains all those tasks the system has chosen for execution.
- single intention is a top level plan instance, plus sub-plans.
- intention structure can contain a number of such intentions (partial order).
- agent is committed to achieve goals, but may reconsider commitments.

Connects the search part of the course (plan and intention) with the reasoning part (beliefs and desires)(1 point).

[7]