Paper Number(s): **E3.16**
ISE3.23

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2002

EEE/ISE PART III/IV: M.Eng., B.Eng. and ACGI

ARTIFICIAL INTELLIGENCE

Thursday, 9 May 10:00 am

There are SIX questions on this paper.

Answer FOUR questions.

Corrected Copy

Time allowed: 3:00 hours

Examiners responsible:

First Marker(s): Pitt,J.V.

Second Marker(s): Shanahan,M.P.

Special Information for Invigilators:

None.

Information for Candidates:

The Prolog General Graph Search Program (GGS)

```
/* search( +Paths, ?Path ) succeeds when
Path is an extension of some path in Paths to a goal
*/

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 ).
```

1 (a) Compare and contrast four algorithms known to you for uninformed search of a problem space.

[12]

(b) Consider the General Graph Search program (GGS).
For each of the algorithms in part (a), specify, in Prolog or other declarative notation, the relations for `choose` and `add_to_paths`. Ensure that any other relations required are also specified.

[8]

2 (a) Define what is meant by an admissible heuristic and a monotonic function. Explain why admissibility and monotonicity are important in A* search. Explain how the path cost function could be made monotonic if the heuristic function was non-monotonic.

[4]

(b) Give two admissible heuristics for path searching in grid-like mazes, and explain why they are admissible.

[4]

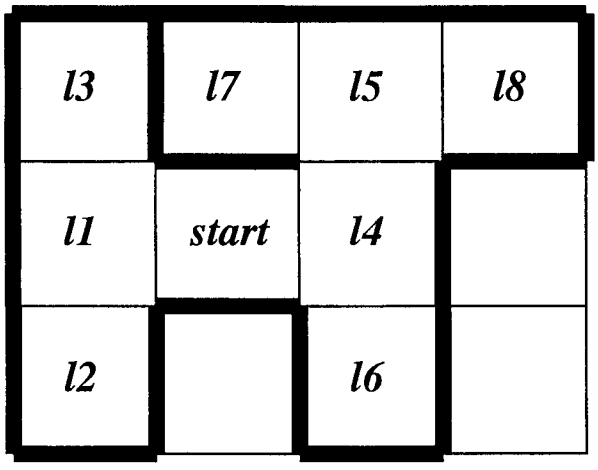
(c) In general, given two admissible heuristics for an A* search, explain why one may be more 'efficient' then the other.

[6]

(d) Explain why, although there may be several solutions to a problem, the first solution found by A* search must be optimal.

[6]

3 Consider the following grid maze, consisting of multiple T-junctions and a robot that is capable of following either the left wall or the right wall, from the *start* location to any of the locations marked *lN* ($1 \leq N \leq 8$):



(a) Give a formal definition of a graph, and illustrate your answer using the maze shown.

[4]

(b) Give an inductive definition of the paths in the graph using the definition in part (a). Illustrate your answer using the maze shown.

[4]

(c) Given a pair of state transformers *left* and *right* with obvious effects, give an inductive definition of a graph based on the root node and the state transformers.

[4]

(d) Hence, or otherwise, give an alternative definition for the paths in a graph. Show how the General Graph Search program (GGS) constructs these paths using breadth first and depth first search. Use the example maze to illustrate your answer.

[8]

4 (a) What is meant by a *conceptualization* in Knowledge Representation. Give different ways in which colour could be conceptualized, for example in a blocks world with differently coloured blocks.

[3]

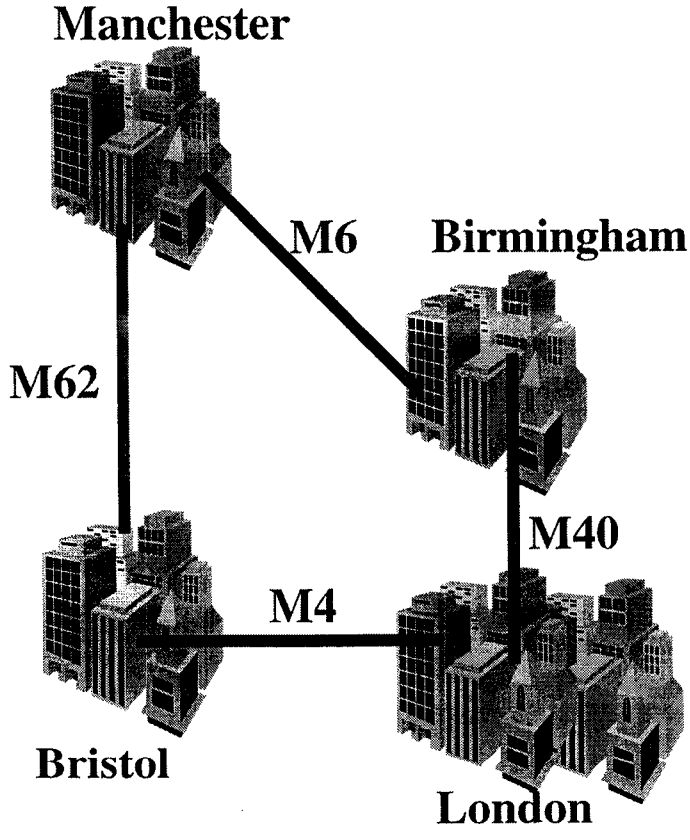
(b) Define what is meant by *unification* in logic programming. Briefly describe a unification algorithm for two terms.

[4]

(c) Describe the inference rule *resolution*, and show that it is sound.

[4]

(d) Consider the following simple map of cities connected by motorways.



(i) Define the facts shown using a relation *connects* only;

(ii) Define rules for the relation *accessible* which is true if there is a route (by motorway or sequence of motorways) from one city to another;

(iii) Show the facts and the rules of parts d(i) and d(ii) as Horn clauses (with the rules implicitly universally quantified);

(iv) Using unification and resolution, show that Manchester is accessible from London.

[9]

5 (a) Explain why $(a \rightarrow c)$ is a correct inference from $(a \rightarrow (b \rightarrow c))$ and b.

[2]

(b) Prove that $(a \rightarrow (b \rightarrow c)) \leftrightarrow ((a \wedge b) \rightarrow c)$. Use the KE calculus and annotate the steps in your proof.
Explain the relation between this proof, the 'proves' relation $\vdash$ between a set of formulas, and proof using the KE calculus in general.

[10]

(c) If a football team wins a match, then it does not lose or draw that match, and vice versa.
Formalize this relationship in propositional logic.
Use this to show that if a team didn't lose, then it either won or drew. Use the KE calculus and annotate the steps in your proof.

[8]

6 (a) Draw the BDI (Beliefs-Desires-Intentions) architecture for a 'rational agent'. State the principal function of each component, and briefly indicate how they interact with each other during one cycle of the interpreter execution cycle.

[4]

(b) For a BDI agent as described in part (a), give a logical formulation and an English description of:
(i) axiom schemas combining the beliefs and desires of a BDI agent that trigger the intention to perform an *inform*, *query*, or *command* communicative act (in which case the agent is the *sender* of a message);
(ii) axiom schemas for the change in belief state of a BDI agent which 'observes' an *inform*, *query* or *command* communicative act (in which case the agent is the *receiver* of the message).

[4]

(c) Illustrate how the axioms of two agents interact when the first agent wants (desires) to know the truth of some proposition p and believes that the second agent knows whether or not p is true. The second agent believes p is not true. Indicate how the desires (goals) of each agent are discharged.

[4]

(d) Specify, in an appropriate notation, a contract net protocol for one agent (the 'client') wanting to allocate a task to one of a group of other agents (the 'workers').
Specify rules which the client and worker agents should observe when negotiating a contract using this protocol.

[8]

E3.16 Artificial Intelligence

Examiners **Dr J. V. Pitt**

First Marker **Dr J. V. Pitt**

Second Marker **Dr. M. P. Shanahan**

There are SIX

MO

?? master - scrn's

check w JVP

ANSWER 1

MARKING SCHEME
(a) 12 marks:
(b) 8 marks:

(a)

	method	time compl	space compl	optimal	complete
depth first	expand next node from deepest level	O(b^m)	O(b*m)	no	no
breadth first	expand next node from shallowest level	O(b^d)	O(b^d)	yes	yes
uniform cost	expand next node with the 'cheapest' cost so far	O(b^d)	O(b^d)	yes	yes
beam	only keep beam width paths of the search space	O(w*d)	O(w*m	no	no

(b)
depth first

```
choose( H, [H|T] T ).

append( New, Other, All ).
```

breadth first

```
choose( H, [H|T] T ).

append( Other, New, All ).
```

uniform cost

```
choose( H, [H|T] T ).

insert_in_order( New, Other, All )

insert_in_order( [], L, L ).
insert_in_order( [H|T], Other, All ) :-
        insert( H, Other, Temp )
        insert_in_order( T, Temp, All ).

insert( H, L, HinL ) :-
        append( Front, [N|Back], L ),
        cost_of( H, Ch ), cost_of( N, Cn ),
        Ch < Cn.
insert( H, L, HinL ) :-
        append( L, [H], HinL ).
```

beam search

```
choose( H, [H|T] T ).

insert_in_order( New, Other, All )
restrict( All, BeamWidth, RestrictedAll )

restrict( All, BeamWidth, All ) :-
        length( All, Lall ),
        Lall =< BeamWidth.
restrict( All, BeamWidth, RestAll ) :-
        append( Fr, _, RestAll ),
        length( Fr, Beamwidth ).
```

ANSWER 2

MARKING SCHEME
(a) 4 marks:
(b) 4 marks:
(c) 6 marks:
(d) 6 marks:

(a)
admissible: never over-estimates
monotonic: uniformly increasing or decreasing.
can't guarantee optimality if g(succ(n)) + h(succ(n)) < g(n) + h(n)
use pathmax equation f(succ(n)) = max(f(n), f(succ(n))

(b)
straight line simple pythag
manhattan, x horizontal + y vertical, must move at least x+y grids

(c)
Let f* be the actual cost of getting to goal node G.
Then A* algorithm
expands all nodes such that g(n) + h(n) < f*
expands some nodes such that g(n) + h(n) = f*
expands _no_ nodes such that g(n) + h(n) > f*

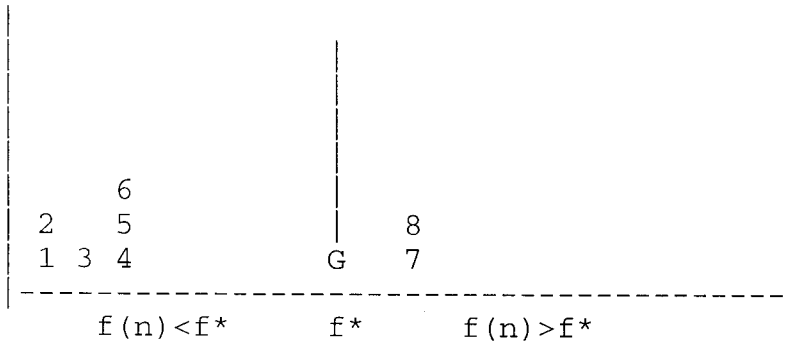
This means A* may expand some nodes for which g(n) + h(n) <= f*, when the actual cost of getting to the goal from n is > f*

For these nodes where actual(n) > f* we want the following inequality to hold:

f* < f(n) + g(n) < actual(n)

i.e we still don't overestimate

If we could create 'histogram' by mapping nodes onto costs;



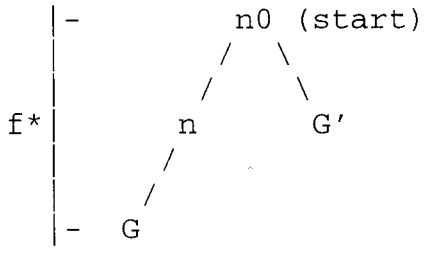
we see that f* expands all those to the left of the bar, some of those on the bar, and none of those to the right of the bar.
A more 'efficient' heuristic will map more of the nodes to the right of the bar.

(c)
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



Then:	f*	≥	f(n)	monotonicity
	f(n)	≥	f(G')	otherwise A* expands n first
	f*	≥	f(G')	transitivity of [3]
	f*	≥	g(G')	a contradiction

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

ANSWER 3

MARKING SCHEME
(a) 4 marks:
(b) 4 marks:
(c) 4 marks:
(d) 8 marks:

(a)
G = < N, R > where
N is the nodes
R is the incidence relation

N = start, l1, l2, l3, l4, l4, l6, l7, l8
R = { (start,l1), (start,l4), (l1,l2), (l1,l3), (l4,l5), (l4,l6), (l5,l7), (l5,l8) }

(b)

$P_G = \bigcup_{i=0}^{\infty} P_i$
where
P_0 = { <start> }
P_{i+1} = { p_i ++ <n_{i+1}> | $\exists\ p_i \in P_i$. (frontier(p_i), n_{i+1}) $\in$ R }

where ++ is append
frontier function gives last node in a sequence

P0 = { <start> }
P1 = { <start,l1>, <start,l4> }
P2 = { <start,l1,l2>, <start,l1,l3>, <start,l4,l5>, <start,l4,l6> }
P3 = { <start,l4,l5,l7>, <start,l4,l5,l8> }

(c)
G = < start_node, Op > where
start_node is the root node
op is set of state transformers

In this case G = < <start>, {left,right} >

where

left(start) = l1	right(start) = l4
left(l1) = l2	right(l1) = l3
left(l4) = l5	right(l4) = l6
left(l5) = l7	right(l5) = l8

and undefined everywhere else

This gives an inductive definition of the nodes and incidence relation of the graph

$N_G = \bigcup_{i=0}^{\infty} N_i$
where
$N_0 = \{ \langle start \rangle \}$
$N_{i+1} = \{ n_{i+1} \mid \exists op \in Op . \exists n_i \in N_i . n_{i+1} = op(n_i) \}$

$R_G = \bigcup_{i=1}^{\infty} R_i$
where
$R_1 = \{ (n_0, n_1) \mid \exists op \in Op . n_1 = op(n_0) \}$
$R_{i+1} = \{ (n_i, n_{i+1}) \mid \exists op \in Op . \exists n_i \in N_i . n_{i+1} = op(n_i) \}$

(d)

$P'_G = \bigcup_{i=0}^{\infty} P'_i$
where
$P'_0 = \{ \langle start \rangle \}$
$P'_{i+1} = \{ p_i ++ \langle n_{i+1} \rangle \mid \exists op \in Op . \exists p_i \in P'_i . n_{i+1} = op(frontier(p_i)) \}$

$P'_0 = \{ \langle start \rangle \}$
$P'_1 = \{ \langle start,l1 \rangle, \langle start,l4 \rangle \}$
$P'_2 = \{ \langle start,l1,l2 \rangle, \langle start,l1,l3 \rangle, \langle start,l4,l5 \rangle, \langle start,l4,l6 \rangle, \}$
$P'_3 = \{ \langle start,l4,l5,l7 \rangle, \langle start,l4,l5,l8 \rangle, \}$

i.e. the inductive definition gives us the same set of paths as before

Now using breadth first, the GGS creates all the paths in each P'_i before searching any of the paths in .

Using depth first search, it picks one path in the deepest P'_i, and computes all those members of P'_{i+1} which are one step extensions of that path, and carries on from there.

In this way, we can search a graph by computing it rather than exploring it if we were given the full explicit definition.

ANSWER 4

MARKING SCHEME
(a) 3 marks:
(b) 4 marks:
(c) 4 marks:
(d) 9 marks:

(a)
Conceptualisation = representation of knowledge in declarative form
formally 3-tuple <domain, functional basis, relational basis>

colour as element of domain	colour(block, red)
colour as function	colour(block) = red
colour as relation	red(block)

(b)
unification = process by is computed a set of substitutions (values for variables) that makes two terms the same

algorithm to unify two terms, X, and Y:
if X is a variable and Y is a variable, then they unify, substitute for each other
if X is a term, and Y is a variable, then they unify, substitution is Y <- X
if X is a variable, and Y is a term, then they unify, substitution is X <- Y
if X and Y are simple terms (constants), then they unify if they are identical
if X and Y are compound terms, then they unify if:
    their functors are the same
    they have the same number of arguments
    each pairwise corresponding argument unfies

(c)
resolution: rule of inference: from $p \vee q$ and $\neg p \vee r$, infer $q \vee r$
resolving p and $\neg p$ is a contradiction
more general case:

To show it is sound, draw up a truth table, and show that when the premises are true, so is the conclusion

(d) showing just the relevant facts:
(i)
connects(*m*4, *london*, *bristol*)
connects(*m*40, *london*, *birmingham*)
connects(*m*62, *bristol*, *manchester*)
connects(*m*6, *birmingham*, *manchester*)

and vice versa, of course.
(ii)
$\forall a, \forall b, \forall m.\ connects(m, a, b) \rightarrow accessible(a, b)$
$\forall a, \forall b, \forall c, \forall m.\ connects(m, a, b) \wedge accessible(b, c) \rightarrow accessible(a, c)$

(iii)
facts as before

$connects(m, a, b) \rightarrow accessible(a, b)$
$= \neg connects(m, a, b) \vee accessible(a, b)$
call this clause 1

$connects(m, a, b) \wedge accessible(b, c) \rightarrow accessible(a, c)$
$= \neg(connects(m, a, b) \wedge accessible(b, c)) \vee accessible(a, c)$
$= \neg connects(m, a, b) \vee \neg accessible(b, c) \vee accessible(a, c)$
call this clause 2

(iv)
goal is: *accessible*(*london*, *manchester*)
query is ¬ *accessible*(*london*, *manchester*) so by resolution and unification:

¬*accessible*(*london*, *manchester*)
    unify with head of clause 2
¬*connects*(*m*4, *london*, *b*) ∨ ¬*accessible*(*b*, *manchester*)
    subst a=london and c=manchester
    unify with fact
    subst b=bristol
¬*accessible*(*bristol*, *manchester*)
    unify with head of clause 1
    subst a=bristol and b=manchester
¬*connects*(*m*, *bristol*, *manchester*)
    unify with fact
    *subst m=m*62
empty goal = success

ANSWER 5

MARKING SCHEME
(a) 2 marks:
(b) 10 marks
(c) 8 marks

(a)
if b is true, $b \rightarrow c$ depends only on value of c, so value of whole thing is $(a \rightarrow c)$

(b)

$\neg((a \rightarrow (b \rightarrow c)) \leftrightarrow ((a \wedge b) \rightarrow c))$	1	*negated conclusion*
branch 1		
$\neg(a \rightarrow (b \rightarrow c))$	2	*PB*,1
$(a \wedge b) \rightarrow c$	4	*e*,1,2
a	5	*a*,2
$\neg(b \rightarrow c)$	6	*a*,2
b	7	*a*,6
$\neg c$	8	*a*,6
$\neg(a \wedge b)$	9	*b*,4,8
$\neg b$	10	*b*,7,9
X	11	*close*,7,10
branch 2		
$a \rightarrow (b \rightarrow c)$	3	*PB*,1
$\neg((a \wedge b) \rightarrow c)$	12	*e*,1,3
$a \wedge b$	13	*a*,12
$\neg c$	14	*a*,12
a	15	*a*,13
b	16	*a*,13
$b \rightarrow c$	17	*b*,3,15
c	18	*b*,16,17
X	19	*close*,14,18

$S \vdash p$
$S' \vdash p$ *where* $S' = \wedge S$
$\vdash S' \rightarrow p$

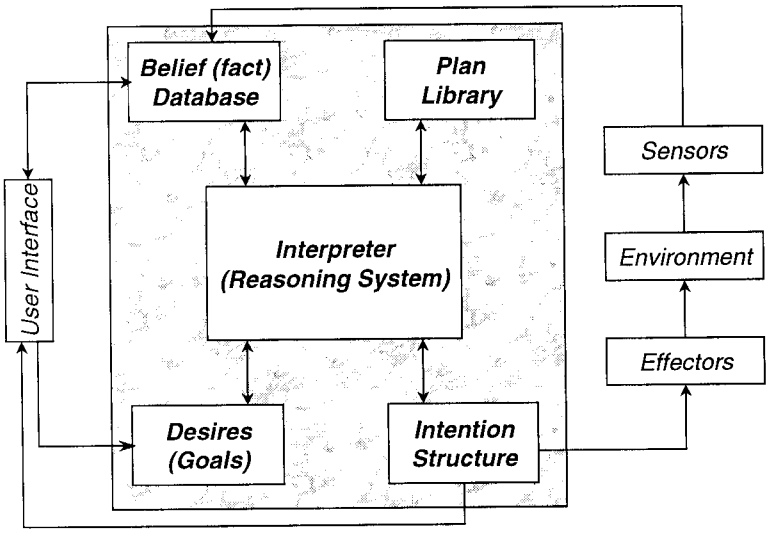
(c)
$win \leftrightarrow \neg lose \wedge \neg draw$ *or*
$win \leftrightarrow \neg(lose \vee draw)$

$win \leftrightarrow \neg(lose \vee draw)$	1	*premise*
$\neg lose$	2	*premise*
$\neg(win \vee draw)$	3	*negated conclusion*
$\neg win$	4	*a*,3
$\neg draw$	5	*a*,3
$\neg\neg(lose \vee draw)$	6	*b*,1,4
$lose \vee draw$	7	$\neg\neg$6
$lose$	8	*b*,5,7
X	9	*close*,2,8

ANSWER 6

MARKING SCHEME
(a) 4
(b) 4
(c) 4
(d) 8

(a)



Main Components
- belief database: facts about the 'world'
- desires: goals to be realized
- plan library: sequences of actions to achieve goals
- intentions: those plans chosen for execution
- interpreter: executes intentions, updates beliefs, modifies goals chooses plans

Interpreter Execution Cycle
- At time t: certain beliefs are held, goals are established, plan (partially) executed
- Events occur: either sensed in environment, or via communication, which alter beliefs or modify goals
- Combination of goals and beliefs will trigger action plan(s)
- One or more will be chosen and put on intention structure
- Interpreter selects an executable plan and executes one step
- Cycle repeats

(b)
Let knowledge K be a shorthand for: $K_s p \leftrightarrow B_s p \vee B_s \neg p$

$\models B_s p \wedge D_s B_r p \rightarrow I_s\langle inform(r,p)\rangle$
s believes p and wants (desires) r to believe p, then s intends to inform r of p

$\models D_s K_s p \wedge B_s K_r p \rightarrow I_s\langle query(r,p)\rangle$
s wants to know p and believes that r knows p, then s intends to query r about p

$\models D_s DONE(A) \wedge B_s Capability(r,A) \rightarrow I_s\langle command(r,A)\rangle$
s wants action A done and believes r is capable of A, then s intends to command r to do A

$[s,inform(r,p)]\ B_r p$
after s informs r that p, r believes p

$[s,query(r,p)]\ (B_r p \rightarrow D_s B_r p) \otimes (B_r \neg p \rightarrow D_r B_s \neg p)$
after s queries r about p, then either r believes p and intends to inform s that it is true, or r does not believe p and intends to inform s that it is false, or,

$[s,command(r,A)]\ I_r\langle A\rangle \wedge (B_r DONE(A) \rightarrow D_r B_s DONE(A))$
after s commands r to do A, then r should intend to do A, and after DONE(A) become true, r should intend to informs that DONE(A) is true

(c)
belief-desire intention states:

sending agent s:

beliefs	$B_s K_r p$		
desires	$D_s K_s p$	*trigger* 2 *gives*	$I_s\langle\langle s, query(r, p)\rangle\rangle$
intentions	{}		

receiving agent r (say):

beliefs	$B_r \neg p$		
desires	{}	*tropism* 2 *gives*	$D_r B_s \neg p$
intentions	{}		

receiving agent r now:

beliefs	$B_r \neg p$		
desires	$D_r B_s \neg p$	*trigger* 1 *gives*	$I_r\langle\langle r, inform(s, \neg p)\rangle\rangle$
intentions	{}		

sending agent s:

beliefs	$B_s K_r p$		
desires	$D_s K_s p$	*tropism* 1 *gives*	$B_s \neg p$ which discharges desire
intentions	{}		

(d)
Can use either FSD or AUML

need to specify:

client does cfp to N workers (for bids for task T within timeout)
client gets M (< N) replies, of which
    i are rejects,
    j are propose, received within timeout
    k are propose, received after timeout
        client issues k rejects, one to each late bidder
client selects winning bid from j valid bids
client sends
    accept to winner
    j-1 rejects. one to each loser
winner worker does task
winner informs client T is done
client pays winner for T

rules are:
ought to be the case that, if client does cfp, client can pay for it
ought to be the case that, if worker does propose, worker can do task
if there is a contract, then ought to be the case that worker is obliged to do task
if there is a contract and task is done, ought to be the case that client is obliged to pay
when the client sends accept to winner, there is a contract

$D_s(E_c cfp(w,T,cnp) \rightarrow Can_c E_c pay(w,T)$
$D_s E_w propose(c,T,cnp) \rightarrow Can_w E_w T$
$E_s contract(c,w,T) \rightarrow D_s O_w E_w T$
$E_s contract(c,w,T) \rightarrow D_s O_c E_c pay(w,T)$
$E_c accept(w,T) \Rightarrow_s E_s contract(c,w,T)$

where s is the institution regulating protocols
Ds is instituional constraint
Can is physical possibility
O is obliged
E is sees to it that
=>s is counts as in s (conditional connective)

THIS PAGE IS OTHERWISE BLANK AND SIGNIFIES END OF THE EXAM SCRIPT AND MODEL ANSWERS FOR
E3.16 ARTIFICIAL INTELLIGENCE.