UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

BSc Honours Degree in Mathematics and Computer Science Part III
MSci Honours Degree in Mathematics and Computer Science Part III
MSc Degree in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the
Diploma of Membership of Imperial College
Associateship of the Royal College of Science

PAPER M3.26

ARTIFICIAL INTELLIGENCE

Tuesday, April 22nd 1997, 2.00 - 4.00

Answer THREE questions

For admin. only: paper contains 4 questions

1a Consider the following, partly defined, fault finder program which uses symptoms to indicate possible faults in components of some device, such as a car or television. It returns a probability of the fault computed as the product of the *strength* with which some *single* symptom correlates with the fault and the *degree* to which the symptom is present.

```
possible_fault_with(Component,Device,Probability):-
   part_of(Component,Device),
   symptom_for_fault_in(Component,Symptom,Strength),
   present(Symptom,Degree),
   Probability is Strength*Degree.
```

i) Which of the, as yet, undefined predicates would need to be defined by a domain expert, and which would need to be defined by facts supplied by a user, in order for this program to become an expert system?

ii) Assume that there can be several symptoms indicating the same fault in a component. Each symptom indicates the fault with a different strength and can be present to a different degree. Modify the above rule for `possible_fault_with/3` so that it computes the probability of a fault as the average `Strength*Degree` for *all* the symptoms for the fault. Assume that there is a predefined relation `average(L,A)` that will compute the average A of a list of numbers L.

b Suppose that the rules of the above program, extended by suitable expert supplied facts and rules for some domain, are to be interpretively executed using the following Prolog query interpreter:

```
eval((Cond,Conj)) :-
   !,eval(Cond), eval(Conj).
eval(Cond) :-
   system(Cond), !,Cond.
eval(Cond) :-
   askable(Cond),!, query_the_user(Cond).
eval(Cond) :-
   clause(Cond,Body), eval(Body).
```

`query_the_user/1` must be defined in such a way that the user is only asked once about each different `askable(Cond)` encountered during the query evaluation. It queries the user by asking them if `Cond` (or some instance of `Cond`) is true. If true, the user can optionally supply a matching fact `F`. The call fails if the user indicates `Cond` is not true, otherwise it succeeds, with any variables it contains bound by the unification `Cond=F`, if the user supplies a matching fact.

i) Give a suitable `askable/1` fact to be added to the above program.

ii) Give a suitable Prolog program for `query_the_user/1`. You can assume the existence of another program `answer_for(Cond)` which will display to the user a suitable dialogue containing the condition `Cond` with buttons for giving the true/false response and an input field for supplying the optional fact `F` in the case of a true response. The call `answer_for(Cond)` will fail if the user hits the false button, will succeed if they hit the true button. It will also unify `Cond` with the user entered fact F, if they give one.

2a i) What is meant by saying that a search algorithm for finding a minimum cost solution path in a problem solving search graph is *admissible*?

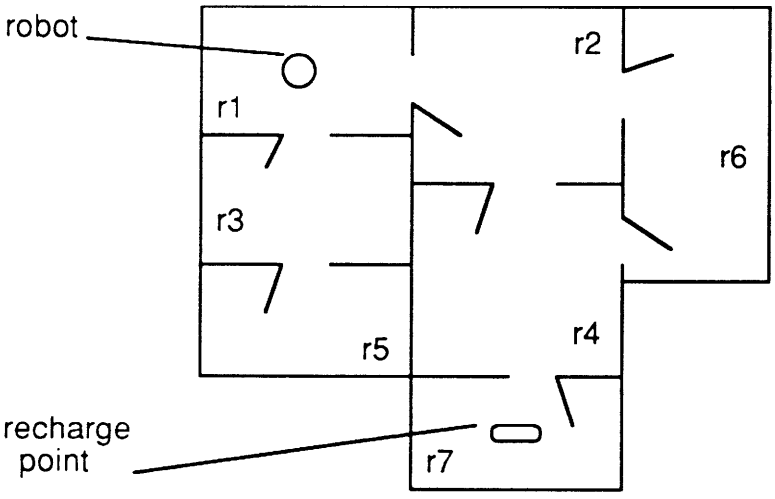
ii) What value is associated with each node N of the exploration of a search graph by the A* graph search algorithm? What node does A* select for expansion at each step in the construction of its search tree.

iii) What are the two conditions on the use of the A* algorithm that guarantee admissibility? Using one of these conditions, prove that any solution path found by A* will be a minimum cost path.

b Suppose a robot has to get to a room that has the recharge point in it and that it decides to find the minimum cost route from its initial position to the recharge point, before it starts out, by using the A* algorithm. It has access to data telling it in which room it is initially located, in which room the power supply is located, and which rooms have connecting doors. It also has access to two tables, one giving the distance between the mid-points of each pair of adjacent rooms for which there is a connecting door, the other giving the straight line distance from the center of any room to the recharge point. The straight line distance may go through one or more walls. In searching for the recharge point the robot can move from the room in which it is located to any adjacent room for which there is a connecting door. The cost of such a move is the distance between the mid-points of the two rooms.

i) How would you compute the numerical value attached to each node in order to apply the A* algorithm to this search problem?

ii) For the room layout and initial state represented by the picture:



draw the search graph of all the rooms that the robot might be in after 1,2 and 3 actions of moving from the middle of one room to the middle of an adjacent room, through a connecting door.

Turn over .....

iii) Separately draw the sub-tree of this graph that will be generated by application of the A* algorithm using your heuristic function. On the search tree label each arc connecting one room to another with the step in the A* algorithm at which the arc will be added to the tree. Thus, all the arcs descending from the initial room r1 will be numbered 1. Assume that A* checks for repeated rooms and prunes them from the search tree. Attach to each room on your search tree the value of the room as it would be computed by the A* algorithm. Assume that the distance between the mid-points of each pair of connected rooms is the same, namely 1, and that the straight line distance from the mid-points of each room to the power point is as given by the following table:

Room	Straight Line Distance to Power Point
r1	2.7
r2	2.0
r3	1.8
r4	1.0
r5	1.1
r6	1.9

3 Assume that the Department of Computing allocates lecturers to courses (one lecturer per course) using the following stepwise recipe:

Stage1: Every lecturer is allocated one introductory course, on a topic for which they have teaching expertise, where the course has an expected enrolment of over 60. There are always enough courses with enrolments over 60 for every lecturer to be given a course during this stage.

Stage2: Every lecturer is allocated exactly one advanced course with a topic that is one of their research areas. Only one advanced course for a given topic is assigned to a lecturer during this stage. Not every lecturer may be assigned a course in this stage.

Stage3: Each lecturer with only one assigned course is assigned a second introductory course on a topic for which they have teaching expertise.

Stage4: Remaining unassigned introductory courses are uniformly allocated to lecturers. Remaining unassigned advanced courses are recorded as withdrawn.

a Write a set of condition/action production rules to implement this recipe. You can assume that the initial data base/working memory includes facts for the relations:

course(C,E,T):	C is a course with expected enrollment E on topic T
adv_course(C,T) :	C is an advanced course on topic T
research_area(L,A) :	A is a research area of lecturer L.
teaching_exp(L,E):	E is an area of teaching expertise for lecturer L.

It also includes the fact: stage1
and a fact: teaches(L,0) for each lecturer L.

The final state of the working memory should include facts for the relations:

taught_by(C,L) :	course C is taught by lecturer L
withdrawn(C):	course C is withdrawn
teaches(L,N) :	lecturer L teaches N courses

Use any production rule notation with which you are familar, but assume that the following are allowed actions:

add(F): fact F is added to the working memory
remove(F): fact F is deleted from the working memory

b State what conflict resolution strategy you are using.

The two parts of this question carry repectively 85%, 15% of the marks.

Turn over .....

4a Briefly describe the following *frame* knowledge representation concepts, and for each one give an example and mention a related concept of object oriented programming.

i) a frame definition
ii) a frame instance
iii) an inherited value for a frame slot
iv) a slot value expression

b

i) Represent the following information using some suitable frame specification notation.

A person has a name (default unknown), a sex (default male), a date_of_birth (default unknown), a father who is a person, a mother who is a person, and a set of siblings (default {}). A person also has an age which is computed, when accessed, by invoking a predefined function `age_of` with argument their date of birth.

A citizen is a person with a nationality. The default value of the nationality of a citizen is the nationality of their father if they are male, else the nationality of their mother. A married citizen is a citizen with a spouse. The default sex for a married citizen is female if the sex of the spouse is male, it is male if the sex of the spouse is female.

The age of a person can only be increased by 1. The father and mother of a citizen must also be citizens. The siblings of a person must all be persons and each must have have the same father or mother as the person.

A citizen P3 has name "B. Smith", father P1, mother P2. P1 has British nationality and P2 is female with Canadian nationality. Married citizen P4 has name "S. Smith", spouse P3, and nationality Irish.

ii) What would the answers be if the frame representation of the above information was queried to find:

1) the nationality of P3?

2) the sex of P4?

and say briefly how each answer will be computed.

The two parts of this question carry repectively 40%, 60% of the marks.

End of paper