UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2001

MEng Honours Degree in Information Systems Engineering Part IV
MEng Honours Degrees in Computing Part IV
MSc in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C474=I4.8

MULTI-AGENT SYSTEMS

Tuesday 8 May 2001, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 a i) Give an abstract functional description of an agent interacting with its environment.

Give functional descriptions of the special classes of:

ii) tropistic agents

iii) hysteretic agents

iv) hysteretic agents with finite memory

b i) What is a mobile agent? Give examples of possible applications.

ii) Give two advantages, and two disadvantages of using mobile agents.

iii) What are the infrastructure requirements for mobile agents and in what forms might a mobile agent be moved.

c i) Briefly describe how you might implement a mobile agent station in Qu-Prolog. *You do not need to give any programs.* How would the agent station store and execute the mobile agent program?

ii) How would you ensure that a mobile agent could only communicate with local agents, could not access the file store of the host machine, and did not consume too much computational resource?

The three parts carry, respectively, 30%, 30% and 40% of the marks.

2a Briefly describe the key features of the *contract net* protocol for allocating tasks to a network of agents/problem solvers. Include descriptions of:

i) the role of a manager

ii) the role of a potential contractor

iii) the content of a contract announcement

iv) when a potential contractor is allowed to submit a bid

v) the use of *node available* messages

vi) the use of contract announcements requesting *immediate response bids* and the possible immediate response replies

b i) Briefly describe an implementation of a contract net agent as a collection of processes in the Qu-Prolog programming language that allows an agent to *concurrently*:

process several tasks

receive and process contract announcements

send out contract announcements

process incoming bids and the announcements it has sent out

You do not need to give any program code.

ii) Briefly explain how you would support the broadcasting of messages to all agents on the net.

The two parts carry, respectively, 60% and 40% of the marks.

3a AGENT0 implements a simple model of a communicating agent.

i) What are the memory components of an AGENT0 agent? What role do they have?

ii) How is an AGENT0 agent programmed? Give a short example.

ii) *Briefly* describe the message formats that the agent can accept, the effect they have on the agent components, and the types of action that an agent can be requested to do.

iii) *Summarise* the execution cycle of the agent.

b i) Sketch how you might implement an AGENT0 style agent in Qu-Prolog. *You do not need to give any programs.*

ii) Suggest generalisations of the AGENT0 model that would be easy to implement in Qu-Prolog.

The two parts carry, respectively, 60% and 40% of the marks.

Turn over...

4a Briefly explain the role of a *matchmaker* or *facilitator* in a distributed information system.

b Explain, with examples of their use, the role of the KQML performatives:

i) advertise
ii) recommend
iii) recruit
iv) subscribe

c The program below is a Qu-Prolog program which, when forked as a publically named process, can act as a simple matchmaker.

```
?-dynamic(has_advertised/2).
match_maker :-
message_choice (
   advertise(ALst) <<- S :: member(content(KQMLMess),AList)  ->
        assert(has_advertised(S,KQMLMess)
    ;
    recommend(AList) <<- S  ::
                member(content(KQMLMess1,AList),
                has_advertised(A,KQMLMess2),
                match(KQMLMess1,KQMLMess2)  ->
                        member(reply_with(L),AList),
                        reply([in_reply_to(L),content(consult(A))]) ->> S),
    match_maker.
```

The program assumes that a KQML message of the form:

(performative :f1 v1 :f2 v2 .... :fk vk)

is represented as the Prolog term:

performative ([f1(v1), f2(v2),..., f(vk)])

and that the :sender, :receiver and :replyto fields are dropped in the Prolog term.

You can assume that match/2 is defined by a suitable Prolog program. Minimally a call to this program will succeed if the two KQML messages being matched have the same performative, mention the same ontology, and have *unifying* content terms.

i) Give the Prolog message term that an agent would send to this matchmaker if it wanted to advertise the fact that it could be asked any query in Prolog using the ontology: dai.

ii) Give the Prolog message that an agent would send to this matchmaker if it wanted to find the identity of an agent to which it could send a Prolog query:

paper_on('KQML',P)

using dai ontology where any reply from the matchmaker should contain the label: lab1.

iii) Give the modifications you would make to the program so that it can also accept and remember key features of KQML style *subscribe* messages. When an agent subsequently advertises with a content message that matches (using the assumed match/2 relation) the content message of some previously received subscription message, a suitable message should be sent, identifying the advertising agent to the subscribing agent, using the reply_with label of the *original* subscription message.

The three parts carry, respectively, 20%, 30% and 50% of the marks