UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1998

BEng Honours Degree in Computing Part III
BSc Honours Degree in Mathematics and Computer Science Part III
MSci Honours Degree in Mathematics and Computer Science Part III
MSc Degree 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
Diploma of Membership of Imperial College
Associateship of the Royal College of Science
Associateship of the City and Guilds of London Institute

PAPER 3.29

LOGIC PROGRAMMING

Tuesday, April 28th 1998, 10.00 - 12.00

Answer THREE questions

For admin. only: paper contains 4 questions

1 The predicate demo(S, T) expresses that from the theory S it is possible to prove the theorem T. The predicate reject(T) expresses that the theorem T is rejected on some basis. The predicate unsatis(S) expresses that the theory S is unsatisfactory exactly because some rejected theorem is provable from it.

a Write a sentence A1 of first-order logic which completely defines unsatis in terms only of demo and reject.

b Assume that demo is transitive, as expressed by

demo(S, T) if demo(S, R), demo(R, T) A2

Write down all the clauses you would start with in order to seek a resolution refutation sufficient to establish that

unsatis(S) if demo(S, R), unsatis(R) A3

is a logical consequence of {A1, A2}.

c Using the clauses identified in part b, construct a resolution refutation from them. Clearly identify the parent clauses used in each resolution step and underline the literals resolved upon.

d Explain, with an illustration, why resolution alone may be unable to derive a refutation from an inconsistent set of clauses when some of those clauses are indefinite. What extra mechanism is needed in such circumstances to ensure a refutation is derivable?

The four parts carry, respectively, 10%, 30%, 50% and 10% of the marks.

2a This program **Q** is defined over the Herbrand domain **H**={a, b, c, d}:

```
arc(a, b).
arc(b, c).
arc(b, d).
arc(c, d).

path(X, Z) if arc(X, Z).
path(X, Z) if arc(Y, Z), path(X, Y).
```

Assuming the Prolog strategy, draw as much of the SLD-tree for the query ?path(X, d) as is necessary to find three distinct solutions.

b The *immediate consequence function* $T_{\mathbf{P}}$ for a definite program **P** is defined by

$$T_{\mathbf{P}}(\mathbf{I}) = \{ q \mid (q \text{ :- body}) \in \mathbf{G}(\mathbf{P}), \text{body} \subseteq \mathbf{I} \}$$

i) Define the minimal model **MM(P)** in terms of $T_{\mathbf{P}}$.

ii) Precisely why is **MM(P)** identical to the set **SS(P)** of atoms that succeed from **P**?

iii) Define the fair finite-failure set **FF(P)** in terms of $T_{\mathbf{P}}$, and explain the significance of the atoms contained in it.

c Use your definitions from parts bi) and biii) to construct **SS(Q)** and **FF(Q)** for the program **Q** shown in part a, making clear the iterates obtained.

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

3a A metalevel interpreter **M** is required for evaluating queries of the form ?demo(X) given some object-level program **P** supplied as data. Each clause of **P** is such that its head is an atomic formula and its body is a conjunction of one or more literals.

Note—a literal is either A or ¬A, where A is any atomic formula, and need not be ground.

Altogether **M** comprises eight definite clauses M1...M8 of which the first three, shown below, constitute the so-called *vanilla* interpreter **V**:

```
demo(true).                          M1
demo(X) if clause(X, Y), demo(Y).    M2
demo(X∧Y) if demo(X), demo(Y).       M3
```

Here, the symbol ∧ denotes object-level conjunction. The predicate clause(X, Y) expresses that the program **P** contains a clause having head X and body Y.

Justify the following statement:

> for any data **P** and any X, Y,
> ?demo(X∧Y) succeeds from **V** iff ?demo(Y∧X) succeeds from **V**

b In order to handle queries containing ¬, **M** also contains

```
demo(¬X) if undemo(X).               M4
```

i) Write down four further definite clauses M5...M8 defining undemo(X) to mean "demo(X) does not succeed from **M**". Use *only* the predicates undemo, forall, clause and demo; do *not* use Prolog's "not" operator.

ii) Draw the tree representing Prolog's evaluation of ?demo(r(X)∧p(X)) using your **M**={M1...M8} and this data **P**:

```
p(X) if ¬q(X).
q(a) if true.
r(b) if true.
```

c Draw the tree representing Prolog's evaluation of ?demo(p(X)∧r(X)) using the same data **P** as in part bii), and hence show that the statement

> for any data **P** and any X, Y,
> ?demo(X∧Y) succeeds from **M** iff ?demo(Y∧X) succeeds from **M**

is not true. Explain carefully the cause of this.

The three parts carry, respectively, 10%, 50% and 40% of the marks.

Turn over ...

4a i) Explain briefly the terms global call-consistency and local call-consistency in relation to the dependency graphs of logic programs.
Which of the two properties implies the other one?

ii) How does the logical consistency of a program's completion relate to whether or not the program is locally or globally call-consistent?

b Given the program **P** with Herbrand domain {bill, hillary, gennifer, monica}:

```
likes(gennifer, bill).
likes(monica, X) if not likes(X, bill).
likes(bill, X) if knows(bill, X).
knows(X, hillary) if likes(X, bill).
```

draw the global and local dependency graphs of **P** and state, with reasons, whether **P** is globally call-consistent, locally call-consistent or neither.

Note—you may use convenient abbreviations in your answers, such as writing lmb in place of likes(monica, bill).

c Write down the completion **Comp(P)**. State any conclusions that can be drawn about its logical consistency in the light of your answers to part b.

d If **Comp(P)** is logically consistent then present a Herbrand model for it, but otherwise present a Herbrand counter-model.

The four parts carry, respectively, 25%, 50%, 20% and 5% of the marks.

End of Paper