UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

BEng Honours Degree in Computing Part III
BSc Honours Degree in Mathematics and Computer Science Part III
MSc Degree in Foundations of Advanced Information Technology
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 City and Guilds of London Institute
Associateship of the Royal College of Science

PAPER 3.29

LOGIC PROGRAMMING—FOUNDATIONS

Tuesday, April 30th 1996, 10.00 - 12.00

Answer THREE questions

For admin. only: paper contains
4 questions
4 pages (excluding cover page)

1a Describe *carefully* how a resolvent R is obtained, in general, by resolving two parent clauses C1 and C2.

What is the logical relationship between R, C1, and C2?

Give an example of a clause C from which two logically-distinct resolvents can be obtained by resolving C with a copy of C, and show what these resolvents are.

b Given a clause-set **C**, further clause-sets $\mathbf{R}_n$ can be defined inductively as follows:

n=0: $\mathbf{R}_n = \mathbf{C}$

n>0: $\mathbf{R}_n = \mathbf{R}_{n-1} \cup \{R \mid R \text{ is a resolvent of any parents in } \mathbf{R}_{n-1}\}$

What is the connection between the consistency of **C** and these further clause-sets? In particular, how does this connection provide an *algorithm* capable of detecting that **C** is inconsistent (when it is)?
Note— you may ignore cases in which factoring would be required.

c Indicate the sets $\mathbf{R}_n$ by which such an algorithm could detect the inconsistency of the clause-set **C** below, identifying clearly the parents of each resolvent:

$$\mathbf{C} = \begin{vmatrix} \text{C1: } \neg\text{path(a, c)} \\ \text{C2: } \text{path(X, Z) if arc(X, Z)} \\ \text{C3: } \text{path(X, Z) if arc(X, Y), path(Y, Z)} \\ \text{C4: } \text{arc(a, b)} \\ \text{C5: } \text{arc(b, c)} \end{vmatrix}$$

*Note— show the **full** contents of $\boldsymbol{R_n}$ **only for** $\boldsymbol{n \leq 1}$.*

The three parts carry, respectively, 25%, 15% and 60% of the marks.

2a The term *SLD-resolution* refers to a restricted form of resolution whereby

- each resolution step employs a *selection function*
- the resolution process is *linear*
- the input clauses are *definite*

Explain *carefully* what these three restrictions actually mean.

Give an example of an inconsistent set of clauses, not all definite, for which non-linear resolution would be needed to demonstrate its inconsistency.

b What is meant by the terms *computation rule* and *search rule* ?
What roles do these rules play in determining and exploring an SLD-tree?
State *briefly* the practical significance of the rules as used in Prolog.

c A hamster is happy if he sees another happy hamster provided he is kept apart from him. Bunjee can see Arthur, who is kept apart from Bunjee. These facts of hamster life are expressed within the Prolog program

```
happy(X) :- apart(X, Y), happy(Y), sees(X, Y).
sees(bunjee, arthur).

apart(arthur, bunjee).
apart(X, Y) :- apart(Y, X).
```

Using this program as written, sketch the SLD-tree determined by Prolog's computation rule for the query ?happy(X), showing the bindings obtained and indicating the part of the tree actually explored.
Note— you may abbreviate predicate and constant symbols as appropriate.

d Propose a better computation rule for the example in part c, and show that your choice *is* better.

The four parts carry, respectively, 20%, 25% , 40% and 15% of the marks.

Turn over ...

3a Given a language **L** based on a set **Pr** of predicate symbols and a set **C** of constant symbols, and which has no function symbols of arity ≥ 1, define *carefully* the following:

i the Herbrand domain **H** determined by **L**,
ii the Herbrand Base **B(P)** over **L** of a definite program **P** all of whose predicate symbols are in **Pr**,
iii a Herbrand interpretation over **L** of this program **P**,
iv the functions $T_P\uparrow\omega$ and $T_P\downarrow\omega$.

b The following program **P**

```
s(X, Y, Z, X, Y, Z)  :-  X<Y, Y<Z.
s(X, Y, Z, U, V, W)  :-  Y<X, s(Y, X, Z, U, V, W).
s(X, Y, Z, U, V, W)  :-  Z<Y, s(X, Z, Y, U, V, W).

1<2.
2<3.
1<3.
```

is to be queried in a language **L** for which **C**={1, 2, 3} and **Pr**={s, <}.
For instance, it might be queried by ?s(3, 2, 1, U, V, W).

i Indicate the contents of **B(P)**.
ii Construct $T_P\uparrow\omega$, showing the iterates obtained.
iii Explain both the model-theoretic and the operational significance of the answer you obtain for $T_P\uparrow\omega$.

c What is the connection, in general, between $T_P\downarrow\omega$ and the notion of a fair computation rule?

For the program **P** in part b, state the contents of $T_P\downarrow\omega$.
Note— do not construct it formally, but explain how you obtained it.

The three parts carry, respectively, 20%, 60% and 20% of the marks.

4a The operational meaning of Prolog's 'not' operator is expressed by the so-called *finite-failure rule*. State this rule precisely.
Construct a simple example, using only propositional (variable-free) Prolog, to show that 'not' is not precisely equivalent to the classical negation $\neg$.

b A propositional sentence of the form (A if (B if C)) can be transformed into two Prolog rules as follows:

```
A :- not E.
E :- C, not B.
```

The following first-order sentence S expresses that a "con-man" is one who, whatever he claims, everyone believes it:

S: $(\forall X)(\text{conman}(X)$ if $(\forall YZ)(\text{believes}(Y, Z)$ if $\text{claims}(X, Z)))$

i Transform S into two Prolog rules, keeping the call-ordering the same as shown in the propositional example. Add to your rules the facts

```
claims(chris, world_is_flat).
believes(bob, world_is_flat).
```

and let **P** denote the entire resulting program.

ii Sketch in full the evaluation of the query ?conman(chris) using **P**, assuming a version of Prolog whose finite-failure rule does not insist that 'not' calls be ground. Make clear what answer is computed for this query.

iii Construct **Comp(P)** on the assumption that the Herbrand domain is just

H={chris, bob, world_is_flat}

and decide (showing your reasoning) whether **Comp(P)** $\models$ conman(chris).

iv Comment on the agreement or otherwise of the answers you obtained in parts ii and iii. Would the use of SLDNF-resolution in part ii have made any difference to the outcome?

Part a carries 15% of the marks.
The subsections of part b carry, respectively, 10%, 25%, 40% and 10% of the marks.

End of Paper