UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

MEng Honours Degrees in Computing Part IV
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 City and Guilds of London Institute

PAPER 4.86

DEDUCTIVE DATABASES

Wednesday, May 7th 1997, 2.00 - 4.00

Answer THREE questions

For admin. only: paper contains 4 questions

1a What is meant by the '*model-theoretic*' and '*proof-theoretic*' approaches to databases? How do the two approaches differ as regards the database itself, queries, and answers to queries?

b i) Construct an example to illustrate that, even with two-valued logics, a deductive database can give *three* possible answers to a closed ('yes/no') query.

ii) Construct an example to illustrate that a database could answer correctly 'yes' to the query $\exists x\, p(x)$? but 'no answer' to the query x: $p(x)$? ('for which x is it the case that $p(x)$?').

c For model-theoretic databases, there is one candidate definition for integrity constraint satisfaction. For proof-theoretic databases there are three candidate definitions. Explain, for both kinds of database, what these definitions are. (Consider integrity constraints of the general form 'If *P* then *Q*'.)

d Construct an example to show that the three candidate definitions for proof-theoretic databases are genuinely distinct. (See also part (e) below.)

e In the case of FOL (first-order logic) databases, what is the possible role of KFOPCE? (It is not necessary to give detailed formal definitions.)

Illustrate your answer by showing how to formulate in KFOPCE your examples from part (d).

2a Transform the following clause into a form to which SLDNF may be applied directly. (You may need to introduce auxiliary predicates.)

$$f(X, W) \leftarrow \forall Y\, [\, (h(Y) \land \forall Z\, [g(X, Z) \rightarrow g(Y, Z)]\,) \rightarrow f(Y, W)\,]$$

b Describe *briefly* the main implementation techniques for *top-down* query evaluation in deductive databases.
Concentrate on identifying the main features and limitations of each approach.

c i) What problem is addressed by the *magic set* technique?
(A *brief* answer is expected.)

ii) Construct the magic set transformation for the following set of 'IDB rules' and the query:

$$p(a, d, Y)\ ?$$

Adorn the query and rules before transforming. *edA* and *edB* are 'EDB' predicates.

$$p(W, X, X) \leftarrow edA(W, X),\ q(X, X)$$
$$p(W, X, Y) \leftarrow edA(Y1, Y),\ p(W, X1, Y1),\ edA(X1, X),\ q(X, Y)$$

$$q(X, Y) \leftarrow edB(X, Y)$$
$$q(X, Y) \leftarrow edB(X, Z),\ q(Z, Y)$$

The three parts carry, respectively, 15%, 35%, 50% of the marks.

3a Let IDB be a set of definite clauses and EDB a set of ground unit clauses.

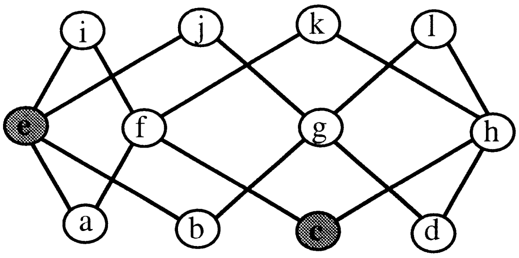
i) Define the closure of EDB under IDB, $Cl_{\text{IDB}}(\text{EDB})$, and state (without proof) its characterisation in terms of Herbrand models and least fixpoints. (It is not necessary to define the immediate consequence operator.)

ii) Present (in any style) the algorithm for *semi-naive* bottom-up evaluation of the 'derived tuples' $s = Cl_{\text{IDB}}(\text{EDB}) - \text{EDB}$.

b Consider the following set of ground unit clauses EDB_1 and definite clauses IDB_1:

EDB_1:

red(*e*)		*red*(*c*)	
arc(*a*, *e*)	*arc*(*b*, *e*)	*arc*(*c*, *f*)	*arc*(*d*, *g*)
arc(*a*, *f*)	*arc*(*b*, *g*)	*arc*(*c*, *h*)	*arc*(*d*, *h*)
arc(*e*, *i*)	*arc*(*f*, *i*)	*arc*(*g*, *j*)	*arc*(*h*, *k*)
arc(*e*, *j*)	*arc*(*f*, *k*)	*arc*(*g*, *l*)	*arc*(*h*, *l*)



(all arcs in the diagram point upwards)

IDB_1:

$$pink(X) \leftarrow red(X)$$
$$pink(X) \leftarrow pink(Y),\ arc(Y, X)$$

Illustrate the algorithm of part (a) by showing the steps of the semi-naive computation of the closure of EDB_1 under IDB_1.

(You may find it helpful to use the diagram to keep track of the computation.)

Indicate in your answer the feature of semi-naive evaluation that distinguishes it from *naive* evaluation.

c Suppose IDB_2 is obtained from IDB_1 by adding the further clause (C_{red}):

$$(C_{red}) \qquad pink(X) \leftarrow arc(X, Z),\ \forall Y[arc(X, Y) \rightarrow red(Y)]$$

Explain how the ABW construction can be used to compute a minimal Herbrand model of $\text{EDB}_1 \cup \text{IDB}_2$. You will need to transform the clause (C_{red}).

There is no need to show the actual computation but you should state clearly any standard results you require.

d Suppose IDB_3 is obtained from IDB_1 by adding, instead of clause (C_{red}), the clause (C_{pink}):

$$(C_{pink}) \qquad pink(X) \leftarrow arc(X, Z),\ \forall Y[arc(X, Y) \rightarrow pink(Y)]$$

Suggest a possible semantics for the database $\text{EDB}_1 \cup \text{IDB}_3$ and discuss *briefly* how it might be computed. There is no need to give definitions of standard terms.

Turn over ...

4a Three of the AGM postulates for revision $K*\alpha$ are:

R3. If $\neg\alpha \notin Cn(K)$ then $K*\alpha = Cn(K \cup \{\alpha\})$

R4. If $\neg\alpha \notin Cn(\emptyset)$ then $K*\alpha$ is consistent

R5. If $Cn(\alpha) = Cn(\beta)$ then $K*\alpha = K*\beta$

Explain the reading of each of these three postulates and of the *Levi identity*

$$K*\alpha = Cn((K - \neg\alpha) \cup \{\alpha\})$$

b Consider the following set of sentences ('database' or 'belief set') of classical (propositional) logic:

$$D = \{ q, r, (p \leftarrow q \wedge r), ((a \wedge b) \vee (\neg a \wedge \neg b)) \}$$

Use D as an example to illustrate the distinction between *revision* and *updating*.

c Use D from part (b) as an example to explain what is meant by the 'view update' problem and its generalisation in deductive databases.

How does the AGM theory contribute to discussions of the view update problem?

d Explain briefly, by means of a suitable example, the difference between 'valid time' and 'transaction time'.

In terms of the AGM theory, what kind of change corresponds to *update* of a *valid time* database?

End of paper

