COMPUTER SCIENCE TRIPOS Part IB

Thursday 6 June 2002 1.30 to 4.30

Paper 6

Answer **five** *questions.*

No more than **two** *questions from any one section are to be answered.*

Submit the answers in five **separate** *bundles, each with its own cover sheet. On each cover sheet, write the numbers of* **all** *attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

SECTION A

1 Data Structures and Algorithms

Arithmetic encoding compactly represents a string of characters by an enormously precise number in the range [0,1) represented in binary by a finite sequence of digits following the decimal point. What is remarkable is that this number can be processed efficiently using only fixed point arithmetic on reasonably small integers. As a demonstration, if the original text contained only the characters `A`, `B`, `C` and the end-of-file mark `w`, such text can be arithmetically encoded using only 3-bit arithmetic. Illustrate how it can be done by decoding the string `101101000010` on the assumption that the character frequencies are such that the decoding tables of size 8 and 6 are, respectively, `wAABBCCC` and `wABBCC`. The first few lines of your working could be as follows:

```
                       0   0   0   0   1   1   1   1
                       0   0   1   1   0   0   1   1
                       0   1   0   1   0   1   0   1
101 101000010        |-w---A---A---B---B-+(C)++C+++C+|       => C
```

Your answer should include a brief description of how the decoding algorithm works.

[20 marks]

2 Computer Design

(*a*) What is *microcode* and how does it differ from assembler? [6 marks]

(*b*) In assembler, branch instructions are used to change the flow of control. How can flow control be determined in a microcode environment? [4 marks]

(*c*) With the aid of a diagram, explain what a *feedback path* (sometimes called a *bypass*) is and how it is used to improve the throughput of a pipeline.

[6 marks]

(*d*) What is a *branch delay slot*? [4 marks]

3 Digital Communication I

Define a *resource* in a digital communication system as anything whose use by one instance of communication prevents simultaneous use by another. Channel capacity is one example.

(*a*) Give *two* more examples of resource in digital communication systems. [4 marks]

(*b*) For the three resources, indicate how the amount of total resource can be increased. [6 marks]

(*c*) How are allocations of each of these resources to instances of communication performed? [10 marks]

4 Concurrent Systems and Applications

A system is to support abortable transactions that operate on a data structure held only in main memory.

(*a*) Define and distinguish the properties of *isolation* and *strict isolation*. [2 marks]

(*b*) Describe *strict two-phase locking* (S-2PL) and how it enforces strict isolation. [4 marks]

(*c*) What impact would changing from S-2PL to ordinary 2PL have (*i*) during a transaction's execution, (*ii*) when a transaction attempts to commit and (*iii*) when a transaction aborts? [6 marks]

(*d*) You discover that the system does not perform as well as intended using S-2PL (measured in terms of the mean number of transactions that commit each second). Suggest why this may be in the following situations and describe an enhancement or alternative mechanism for concurrency control for each:

(*i*) The workload generates frequent contention for locks. The commit rate sometimes drops to (and then remains at) zero. [2 marks]

(*ii*) Some transactions update several objects, then perform private computation for a long period of time before making one final update. [2 marks]

(*iii*) Contention is extremely rare. [4 marks]

SECTION B

5 Comparative Programming Languages

(*a*) Briefly discuss the compromises that must be made when standardising a programming language. [8 marks]

(*b*) Discuss the relative merits to (1) the application programmers and (2) compiler writer of the following ways of specifying a programming language.

(*i*) A concise readable user manual for the language in English containing many useful programming examples.

(*ii*) A very long and highly detailed description, in English, of every feature of the language. This manual contains no programming examples.

(*iii*) A concise but rigorous description using a formal grammar to describe the language syntax and making extensive use of mathematical notations taken from set theory, λ-calculus, predicate calculus and logic to describe the semantics of the language.

(*iv*) The source code for a clean and elegant machine-independent interpretive implementation of the language.

[12 marks]

6 Compiler Construction

Explain a possible implementation technology for Java classes and objects. Your answer should focus on storage layout for objects and on how class variables and methods are accessed—it is not necessary to explain access qualifiers such as `public` and `private`. Illustrate your answer with the following program; in particular indicate its eventual output.

```
class test {
public int n;
public static int s = 100;
public void f(int x) { System.out.println("f1 " + (x+n)); }
public static void main(String args[]) {
        test p = new test();
        test2 q = new test2();
        test r = q;
        p.n = 4;
        q.n = 5;
        q.m = 6;
        r.n = 7;
        p.f(p.s);
        q.f(p.s);
        r.f(q.s);
     }
}

class test2 extends test {
public int n, m;
public static int s = 200;
public void f(int x) { System.out.println("f2 " + (x+n+m)); }
}
```

[20 marks]

7 Prolog for Artificial Intelligence

A simple D-type flip-flop is represented by the Prolog predicate `dff` whose definition is as follows:

```
dff(D, 0, Q, Q).
dff(D, 1, Q, D).
```

The first argument is the input to the flip-flop, the second is the clock with `0` representing a falling edge and `1` representing a rising edge. The third and fourth arguments are the previous and next states of the flip-flop. As can be seen the state of the flip-flop changes on a rising edge of the clock.

A clocked circuit consists of three d-type flip-flops with inputs and states (D_1, Q_1), (D_2, Q_2) and (D_3, Q_3). They are wired in such a way that

$$D_1 = (Q_1 \wedge Q_2) \vee (\overline{Q_1} \wedge \overline{Q_2})$$
$$D_2 = (\overline{Q_1} \wedge Q_3) \vee (Q_2 \wedge \overline{Q_3})$$
$$D_3 = (Q_1 \wedge Q_3) \vee (\overline{Q_2} \wedge \overline{Q_3})$$

(*a*) Using `s(Q1, Q2, Q3)` to represent the state of the circuit, define a predicate that will compute the state after the next rising edge of the clock. You may find it helpful to define predicates to represent *and*, *or* and *not* gates. [14 marks]

(*b*) Define a predicate `testcc(N, s(Q1,Q2,Q3), List)` that will compute the list of states (`List`) through which the circuit passes from the given initial state `s(Q1,Q2,Q3)` as a result of a sequence of `N` rising edges of the clock. [6 marks]

8 Databases

(*a*) Define the operators in the (core) relational algebra. [6 marks]

(*b*) The core relational algebra is often extended with other operators. For the following operators give a definition and an example of their behaviour:

(*i*) the full outer join operator; [3 marks]

(*ii*) the aggregate and grouping operator. [5 marks]

(*c*) X, Y and Z are all relations with a single attribute A. A naïve user wishes to compute the set-theoretic expression $X \cap (Y \cup Z)$ and writes the following SQL query.

```
SELECT X.A
FROM   X,Y,Z
WHERE  X.A=Y.A OR X.A=Z.A
```

(*i*) Give the relational algebra term that this query would be compiled to. [2 marks]

(*ii*) Does the SQL query satisfy the user's expectation? Justify your answer. [4 marks]

SECTION C

9 Semantics of Programming Languages

A call-by-value evaluation relation for λ-terms that are closed (i.e. ones without free variables) is inductively defined by the axiom

(1) $$\lambda x.\, M \Downarrow \lambda x.\, M$$

and the rule

(2) $$\frac{M_1 \Downarrow V_1 \quad M_2 \Downarrow V_2 \quad M[V_2/x] \Downarrow V}{(M_1\, M_2) \Downarrow V} \text{ if } V_1 = \lambda x.\, M$$

where V_1, V_2, V range over closed λ-abstractions and $M[V_2/x]$ denotes the result of substituting V_2 for all free occurrences of the variable x in the λ-term M. A call-by-value *applicative simulation* is a binary relation $\mathcal{S}$ between closed λ-terms satisfying that whenever $M_1\ \mathcal{S}\ M_2$ and $M_1 \Downarrow V_1$, then for some V_2 it is the case that $M_2 \Downarrow V_2$ and $(V_1\, V)\ \mathcal{S}\ (V_2\, V)$ for all V. Write $M_1 \leqslant M_2$ to mean that $M_1\ \mathcal{S}\ M_2$ holds for some such $\mathcal{S}$.

(*a*) Give a closed λ-term Ω with the property that $\Omega \Downarrow V$ holds for *no* V. [4 marks]

(*b*) Deduce that $\Omega \leqslant M$, for all M. [2 marks]

(*c*) Show that $M \leqslant M$, for all M. [2 marks]

(*d*) Show that $M[V/x] \leqslant (\lambda x.\, M)V$, for all M, V, x. [6 marks]

(*e*) Is it always the case that $M[N/x] \leqslant (\lambda x.\, M)N$ holds when N is not a λ-abstraction? [Hint: consider the case when $N = \Omega$ and M is a suitable λ-term not containing x free.] [6 marks]

10 Foundations of Functional Programming

Show how to use pure lambda terms to represent

(*a*) truth values and the ability to make tests on them;

(*b*) tuples;

(*c*) something like the effect of the ML
```
datatype blist = empty | node of bool*blist
```

(*d*) a fixed-point operator;

(*e*) code to reverse a list of boolean values.

[4 marks each]

11 Logic and Proof

(*a*) The formula A is converted to clauses, omitting the initial step of negating the formula. Resolution is attempted, but the empty clause cannot be derived. What can we conclude about A? [3 marks]

(*b*) The formula A is converted to clauses by the usual procedure, except that Skolemization is performed before negating the formula instead of afterward. Resolution is attempted and the empty clause is derived. What can we conclude about A? [3 marks]

(*c*) The formula A is converted to clauses by the usual procedure. The Davis–Putnam method is applied. In some of the case splits the empty clause is derived, but in others it is not. What can we conclude about A? [3 marks]

(*d*) For each of the following sequents, present either a formal proof or a falsifying interpretation. The modal logic is S4.

$$((\exists x\, P(x)) \to Q) \to \forall x\, (P(x) \to Q)$$

$$\Diamond\Box A \to \Diamond\Box\Diamond\Box A$$

[5 + 6 marks]

12 Complexity Theory

(*a*) State and prove the time hierarchy theorem. [10 marks]

(*b*) For each of the following statements, state whether or not it can be derived as a consequence of the time hierarchy theorem. Give justification for your answer.

(*i*) There is a language in $\textsc{Time}(n^2)$ that is not in $\textsc{Time}(n \log n)$. [3 marks]

(*ii*) There is a language in $\textsc{Time}(2^n)$ that is not decidable in polynomial time. [4 marks]

(*iii*) There is a language in $\textsc{Time}(2^n)$ that is not in **NP**. [3 marks]

END OF PAPER