UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1999

MEng Honours Degrees in Computing Part IV
MSci Honours Degree in Mathematics and Computer Science 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 Associateship of the City and Guilds of London Institute

PAPER 4.70

PROGRAM ANALYSIS

Wednesday, May 12th 1999, 10.00 – 12.00

Answer THREE questions

For admin. only:
paper contains 4 questions

1a Consider the following simple functional programming language:

```
e   ::=   t^l

t   ::=   fn x => e | let x = e1 in e2 | e1 e2 |
          e1 * e2 | x | n
```

Write down an abstract specification of a control flow analysis for this language (0-CFA).

b Consider the term:

```
let  f = fn g => fn x => g (g x)
    in  let two = fn y  => y * 2
        in (f two) 3
```

i) Write down the labelled version of the term.

ii) Write down a guess for the analysis result.

iii) Use the abstract specification of part (a) to verify the guess for the sub-expression

```
(g x)
```

c Consider a *Detection of Signs Analysis* using **Data** = {pos, neg, 0} and abstract values:

$\wp$(**Term** $\cup$ **Data**)

i) Write down an abstract version of *.

ii) Change the abstract specification of part (a) to incorporate the data flow analysis. You need only write down the clauses which change.

2a Define the notions of *Monotone Framework* and *instance* of a Monotone Framework. Illustrate your answer using one of the classical data flow analyses (Reaching Definitions, Available Expressions, Live Variables or Very Busy Expressions).

b A parity analysis detects whether the value in a variable is odd or even or undetermined (either odd or even) every time we reach a certain program point. Using

$$(\mathrm{Var}_* \to \{\mathrm{odd}, \mathrm{even}\}^\top)_\perp$$

as the set of abstract states, write down a parity analysis (as an instance of a Monotone Framework) for the following language:

$$a ::= x \mid n \mid a_1 * a_2$$

$$b ::= \ldots$$

$$S ::= [x := a]^l \mid [\mathrm{skip}]^l \mid S_1 ; S_2 \mid$$

$$\mathrm{if}\ [b]^l\ \mathrm{then}\ S_1\ \mathrm{else}\ S_2 \mid$$

$$\mathrm{while}\ [b]^l\ \mathrm{do}\ S$$

c Define the notion of *Distributive Framework.* Is the framework for parity analysis (part (b)) distributive? Justify your answer.

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

Turn Over ...

3a A variant of Reaching Definitions replaces RD $\in \wp(\text{Var} \times \text{Lab})$ by RL $\in \wp(\text{Lab})$: a label should suffice for finding the variable that may be assigned to some elementary block bearing that label. Using the notation $?_x$ to indicate that we have not yet encountered a definition for x, write down the data flow equations and kill and gen functions necessary to analyse programs of the following language:

```
S ::=   [x := a]^l | [skip]^l | S1 ; S2 |
        if [b]^l then S1 else S2 |
        while [b]^l do S
a ::=   ...
b ::=   ...
```

(assume the existence of the functions init, flow and blocks and that programs are label consistent and have isolated entries).

b Consider the following program:

```
[x := 2]^1; (while [true]^2 do [x := x + 1]^3;) [z := x]^4;
```

i) Write down the equations that would be generated by a Live Variables Analysis of this program.

ii) Write down the sequence of approximations to the solution of the equations in part (bi).

c i) What goes wrong if we naively try to apply intra-procedural techniques to languages with procedures?

ii) Give an informal definition of *valid paths.* Why is the MVP solution not a satisfactory solution to the problems encountered in part (ci).

iii) Briefly sketch how the notion of "context" may be used to provide a satisfactory solution.

(The three parts carry, respectively, 40%, 40% and 20% of the marks)

4a Write down the worklist algorithm for equation solving in Monotone Frameworks.

b The constraints generated from Control Flow Analysis can also be solved using a worklist algorithm. Describe a suitable structure for representing the constraints in this algorithm. How do the components of the structure relate to the concepts used in your answer to part (a).

c. Write down a syntax-based specification of a 0-CFA for the language:

$$e ::= t^l$$

$$t ::= \texttt{fn}\ x \Rightarrow e \mid e_1\ e_2 \mid x \mid c \mid e_1 * e_2 \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2$$

which takes account of the left-to-right evaluation order imposed by a call-by-value semantics.