UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2001

MEng Honours Degrees in Computing Part IV
MSc 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 C470

PROGRAM ANALYSIS

Thursday 17 May 2001, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1a i) Give an example to illustrate what goes wrong if we naively try to apply intra-procedural data flow analysis techniques to languages with procedures.

ii) Briefly describe the MVP solution which addresses the problems encountered in part i. Is this a satisfactory solution?

b Consider the following simple functional programming language:

$$
\begin{array}{lll}
\texttt{t} & ::= & \texttt{fn x => e} \mid \texttt{e}_1 \texttt{ where x = } \texttt{e}_2 \mid \texttt{e}_1\ \texttt{e}_2 \mid \\
 & & \texttt{e}_1 \texttt{ + } \texttt{e}_2 \mid \texttt{x} \mid \texttt{n} \\
\texttt{e} & ::= & \texttt{t}^{\ell}
\end{array}
$$

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

c Consider the term:

```
(three inc) 2
   where three = fn f => fn x => f (f (f x))
      where inc = fn y => y + 1
```

i) Write down a labelled version of the term.

ii) Write down a guess for the analysis result.

2 Consider the following simple functional programming language:

```
t ::=  fn (x₁ ...xₙ) => e | e (e₁ ...eₙ) |
       x | c | e₁ + e₂ |
       e₁ where x = e₂
```

where expressions are labelled terms.

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

b The constraints generated from the 0-CFA can be solved using a worklist algorithm. Describe a suitable structure for representing the constraints in this algorithm.

c A parity analysis detects whether the value of an expression is odd or even or undetermined (either odd or even). Choose a suitable representation for this information and extend the control flow analysis of part (a) to perform this data flow analysis.

3 Consider the following simple imperative language with statements of the following form:

$$\begin{array}{rcl} S & ::= & \texttt{x := a} \\ & | & \textbf{skip} \\ & | & S_1 \; ; \; S_2 \\ & | & \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \\ & | & \textbf{while } b \textbf{ do } S \end{array}$$

A Minimum/Maximum Analysis MinMax as an instance of the monotone framework (similar to the Constant Propagation Analysis CP).

> For each program point, the MinMax Analysis will determine the minimal and maximal value of a variable may have, whenever execution reaches that point.

Assume the usual labelling, take the flow, F, to be *flow*$(S_\star)$, and the extremal labels, E, to be $\{init(S_\star)\}$.

a Look at the set of all intervals, including unbound and empty intervals:

$$\textbf{Interval} = \{[min, max] \mid min, max \in \mathbf{Z}^{\infty}_{-\infty}\} \cup \bot$$

where $\mathbf{Z}^{\infty}_{-\infty} = \mathbf{Z} \cup \{\infty, -\infty\}$.
Define a least upper bound operator $\sqcup$ for intervals and a partial order $\sqsubseteq$.
Define interval arithmetic operations $\widehat{+}$, $\widehat{-}$, and $\widehat{*}$, such that if $x \in [min_1, max_1] = I_1$ and $y \in [min_2, max_2] = I_2$ one can guarantee that $x \text{ op } y \in I_1 \; \widehat{\text{op}} \; I_2$.

b Define $\sqcup$ and $\sqsubseteq$ on the set of states:

$$\widehat{\textbf{State}} = (\textbf{Var}_\star \to \textbf{Interval})_\bot.$$

Define the transfer functions for MinMax, f_ℓ, together with an evaluation function:

$$\mathcal{A} \; : \; \textbf{AExp} \to (\widehat{\textbf{State}} \to \textbf{Interval})$$

of an expression a in a state $\sigma \in \widehat{\textbf{State}}$ based on the interval arithmetic operations above $\widehat{+}$, $\widehat{*}$, ...

c Discuss how the information obtained by the MinMax analysis could be used to simplify a program.

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

4 Consider the following simple imperative language with statements of the following form:

$$
\begin{array}{rcl}
S & ::= & \texttt{x := a} \\
 & | & \textbf{skip} \\
 & | & S_1 \ ; \ S_2 \\
 & | & \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \\
 & | & \textbf{case } a \textbf{ of } n_1 : S_1 \mid \ldots \mid n_i : S_i \\
 & | & \textbf{repeat } S \textbf{ until } b \\
 & | & \textbf{while } b \textbf{ do } S
\end{array}
$$

a Define an appropriate labelling for statements/blocks and give a definition for the flow and reverse flow, *flow* and flow^R.

b Define the equation schemes for a Reaching Definition Analysis RD. You should define RD_{entry} and RD_{exit} and all needed auxiliary functions (with their type).

c Use these schemes to analyse the following program (introduce labels, formulate equations, construct the solutions):

```
x := 3;
y := 0;
repeat ( x := x-1 ; y := y+x )
until x == 1;
x := y;
```