UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1998

MEng Honours Degrees in Computing Part IV
MEng Honours Degree in Information Systems Engineering 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
Diploma of Membership of Imperial College
Associateship of the City and Guilds of London Institute
Associateship of the Royal College of Science

PAPER 4.19 / I4.6

ADVANCED COMPILERS

Thursday, May 7th 1998, 2.00 - 4.00

Answer THREE questions

For admin. only: paper contains 4 questions

1a Consider *available expressions*: an expression a+b is available at a point p if on all paths from initial point to p, there exists a statement x=a+b which is not followed (in the path) by an assignment to a or to b.

For any statement S, we distinguish between.

- `S.in` set of expressions available immediately before `S`.
- `S.out` set of expressions available immediately after `S`.
- `S.gen` set of expressions generated by `S`, ie `S` contains a calculation of `a+b` which is not followed (in `S`) by an assignment to `a` or `b`.
- `S.kill` set of expressions killed by `S`, ie `S` contains an assignment to `a` or `b` which is not followed (in `S`) by a calculation of `a+b`.

i) Give the equations for available expressions for the following syntax, in terms of `S.in`, `S.out`, `S.gen` and `S.kill`.

```
G -> S
S -> x := y + z
```

ii) Give the equations for available expressions across basic blocks `B`, in terms of B.in, B.out, B.gen and B.kill.

iii) Does the assignment a:=a+b kill or generate a+b? Justify.

b Consider *very busy expressions*: an expression a+b is very busy at a point p if on all paths from p, the expression a+b is evaluated prior to any assignment to a or to b.

i) For a statement `S`, give the meaning of `S.in`, `S.out`, `S.gen` and `S.kill`, for very busy expressions.

ii) Give the equations for very busy expressions for the following syntax, in terms of `S.in`, `S.out`, `S.gen` and `S.kill`.

```
G -> S
S -> x := y + z
```

iii) Give the equations for very busy expressions across basic blocks `B`, in terms of `B.in`, `B.out`, `B.gen` and `B.kill`.

iv) Does the assignment `a:=a+b` kill or generate `a+b`? Justify.

The two parts carry, respectively, 30% and 70% of the marks.

2 Consider the following context free grammar, which describes literals in relation to a base:

```
p1: Literal   ::= base @ Digits.
p2: Digits    ::= digit Digits.
p3: Digits    ::= .
```

For instance, `5@234` is a literal in this language, and its value is `2*25+3*5+4=69`. In general, the literal $b@x_1x_2...x_n$, where the base, b, is a positive number, and x_1, x_2...x_n are numbers greater or equal to 0 and smaller than b, has the value: $x_1*b^n + x_2*b^{n-1}+...x_n*b^0$.

a Write an attribute grammar to express the value of such literals. You may assume that the terminals **base** and **digit** have each an intrinsic attribute `value` of type `Cardinal`.

i) For each symbol list its attributes, indicating whether they are synthesized or inherited.

ii) For each production, give the attribution rules.

b i) Draw the tree with the attribute dependencies for `8@16`.

ii) To which implementation class does the attribute grammar from part a belong (eg S-AG, LAG(n), etc)?

Turn over ...

3 a Define the notion of *Monotone Framework* and *instance* of a Monotone Framework. Illustrate your answer by one of the classical data flow analyses (reaching definitions, available expressions, live variables or very busy expressions).

b *Sinking* is an optimisation that complements hoisting. It moves an assignment to a common successor, making a number of identical assignments in predecessors redundant. For example in:

```
if a = b
then c := a + b; b := 0
else c := a - b; b := 0;
a := 1
```

the assignment "`b := 0`" can be moved to just after the conditional.

For this optimisation to be valid, it must be the case that on all paths from the original assignment (`x := expr`) to the new assignment:

1. There is no assignment that changes the value of `expr`, and
2. The variable `x` is not referenced.

Define a data flow analysis for this problem.

c Describe an algorithm for solving the flow equations of part b.

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

4 a Consider the simple functional language:

$e ::= t^l$

$t ::= c \mid x \mid \text{fn } x \Rightarrow e \mid e_1\ e_2$

Formulate a syntax-directed specification of 0-CFA for this language.

b Write down the constraints generated by your answer to part a for the following program:

$((\text{fn } x \Rightarrow (x^1\ 4^2)^3)^4\ (\text{fn } y \Rightarrow y^5)^6)^7$

Give the least solution of these constraints.

c Define a combined Control Flow and Data Flow Analysis that performs a *parity analysis*, using Data = {**odd**,**even**}, for the language. The purpose of the parity analysis is to determine whether numeric values that can occur at a point are always odd, even or a mixture.

End of paper