UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BSc Honours Degree in Mathematics and Computer Science Part II
MSci Honours Degree in Mathematics and Computer Science Part II
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 Royal College of Science
Associateship of the City and Guilds of London Institute

PAPER 2.12 / MC2.12

SEMANTICS—OPERATIONAL

Tuesday, May 7th 1996, 4.00 - 5.30

Answer THREE questions

For admin. only: paper contains
4 questions
4 pages (excluding cover page)

1 The following is the abstract syntax of a Turtle control language:

$p \in$ Program

$n \in$ Numeral

$a \in$ Arithmetic-expression

$p ::= up \mid down \mid forward(a) \mid left \mid right \mid p_1 ; p_2$

$a ::= n \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2$

The Turtle navigates its way around a two dimensional space. It has a pen which may be up or down; in the latter case it leaves an ink trace of its movements. It can turn left or right through 90 degrees and it can move forward a distance specified by the parameter.

a i) Define the Natural Semantics of arithmetic expressions.

ii) The state of the Turtle is represented by a quadruple: (x,y,pen,direction). The first two elements give the Turtle's current position (cartesian coordinates); the third element is a boolean indicating whether the pen is up or down; the fourth element gives the current forward direction (North, East, West or South). Use your answer to a(i) to define the Natural Semantics of programs.

b The Turtle has configurations $<c, e, s> \in$ Code $\times$ Stack $\times$ State, where:

$c \in$ Code

$i \in$ Instruction

$c ::= \varepsilon \mid i : c$

$i ::=$ PUSH-n | ADD | SUB | MULT | UP | DOWN | FORWARD | LEFT | RIGHT

Define an operational semantics for the Turtle.

c Define suitable translation functions to translate control programs into Turtle code.

d Assuming that:

if $<c_1, e_1, s> \triangleright^k <c', e', s'>$ then $<c_1 : c_2, e_1 : e_2, s> \triangleright^k <c' : c_2, e' : e_2, s'>$

show that the translation function for arithmetic expressions is correct.

The four parts carry, respectively, 35%, 30%, 10%, 25% of the marks.

2 The abstract syntax for the language Repeat is given by:

x ∈ Variable

a ∈ Arithmetic-expression

b ∈ Boolean-expression

S ∈ Statement

S ::= x := a | S_1 ; S_2 | **if** b **then** S_1 **else** S_2 | **skip** | **repeat** S **until** b

The syntax of expressions is unspecified.

a Define the Natural Semantics of Repeat statements (assuming the existence of suitable functions for the semantics of expressions).

b Extend the syntax of Repeat with an **assert** statement:

... | **assert** b **before** S

The idea is that if b evaluates to true then S is executed and otherwise the execution of the complete program aborts. Extend the Natural Semantics of Repeat to express this. Show that **assert true before** S is semantically equivalent to S.

c Extend the syntax of Repeat with a **case** statement:

... | **case** a **of** a_1 : S_1...a_n : S_n

The intuitive semantics is that the first statement, S_i, such that a = a_i is executed; if there is no such statement, the effect of the **case** is the same as **skip**. Write down both a Natural Semantics and a SOS-style semantics for the new construct.

Turn over ...

3a Explain, with suitable examples, the concepts of *static* and *dynamic* scoping for procedures and variables.

b Define a suitable syntax for declarations and calls of parameterless procedures. Write a Natural Semantics for declarations and non-recursive calls that enforces static binding for both procedures and variables. You may assume that you are provided with a transition relation that gives the semantics of variable declarations.

c Modify the syntax of procedure declarations and calls so that procedures take a single call-by-value parameter. Modify the semantics given as your answer to part (b) to handle this. You will need to use a procedure environment which maps each procedure name to a quadruple consisting of the parameter name, the body, a variable environment and a procedure environment.

4 The abstract syntax for the language Loop is given by:

x ∈ Variable

a ∈ Arithmetic-expression

b ∈ Boolean-expression

S ∈ Statement

S ::= x := a | S_1 ; S_2 | **if** b **then** S_1 **else** S_2 | **skip** |

loop S_1 **exif** b **endx** S_2 **endl**

The syntax of expressions is unspecified.

a Assuming the existence of suitable functions for the semantics of expressions, write down the denotational semantics of statements.

b Use the denotational semantics to compute the meaning of:

loop skip exif x = 0 **endx** x := x - 1 **endl**

c Determine which of the following functionals (mapping partial functions on State to partial functions on State) are monotone:

i. $F\,g = g$

ii. $F\,g = g_1$ if $g = g_2$

g_2 otherwise

where $g_1 \neq g_2$

iii. $(F\,g)\,s = g\,s$ if $s\,x \neq 0$

s if $s\,x = 0$

End of paper