UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

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

Friday, May 9th 1997, 10.00 - 11.30

Answer THREE questions

For admin. only: paper contains 4 questions

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

$$
\begin{array}{lcl}
p & \in & \text{Program} \\
n & \in & \text{Numeral} \\
a & \in & \text{Arithmetic} - \text{expression} \\
p & ::= & \texttt{up} \mid \texttt{down} \mid \texttt{move}(a_1, a_2) \mid p_1;p_2 \\
a & ::= & n \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2
\end{array}
$$

The Snail 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 move from its current position to the relative co-ordinates indicated by the vector (a_1, a_2).

a i) Define the Natural Semantics of arithmetic expressions.

ii) The state of the Snail is represented by a triple: *(x,y,pen)*. The first two elements give the Snail's current position (cartesian coordinates); the third element is a boolean indicating whether the pen is up or down. Use your answer to a(i) to define the Natural Semantics of programs.

b The Snail has configurations $\langle c, e, s\rangle \in \text{Code} \times \text{Stack} \times \text{State}$, where:

$$
\begin{array}{lcl}
c & \in & \text{Code} \\
i & \in & \text{Instruction} \\
c & ::= & \varepsilon \mid i : c \\
i & ::= & \texttt{PUSH}{-}n \mid \texttt{ADD} \mid \texttt{SUB} \mid \texttt{MULT} \mid \texttt{UP} \mid \texttt{DOWN} \mid \texttt{MOVE}
\end{array}
$$

Define an operational semantics for the Snail.

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

d Assuming that:

$$\text{if } \langle c_1, e_1, s\rangle \rhd^k \langle c', e', s'\rangle \text{ then } \langle c_1 : c_2, e_1 : e_2, s\rangle \rhd^k \langle c' : c_2, e' : e_2, s'\rangle,$$

and that the translation function for arithmetic expressions is correct, show that the translation function for programs is correct.

The four parts carry, each, 25% of the marks.

2 The abstract syntax for the language Exif-Loop is given by:

$x \in$ Variable
$a \in$ Arithmetic – expression
$b \in$ Boolean – expression
$S \in$ Statement
$S ::= x := a \mid S_1;S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{skip} \mid \texttt{loop } S_1\texttt{; exif } b\texttt{; } S_2 \texttt{ endl}$

The idea of the `loop - endl` construct is that, other than in a `while` loop, before testing the boolean, first a number of statements will be executed.

The syntax of expressions (both arithmetic and boolean) is unspecified.

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

b Extend the syntax of Exif-Loop with a `repeat` statement:

$$\texttt{repeat } S \texttt{ until } b$$

Extend the Natural Semantics of Exif-Loop to cover this extension. Show that '`repeat` S `until` b' is semantically equivalent to '`loop` S; `exif` b; `skip endl`'.

c Extend the syntax of Exif-Loop with a `times` statement:

$$\texttt{times } a \texttt{ do } S$$

The intuitive semantics is that the statement is executed the number of times specified by the arithmetic expression. Write down both a Natural Semantics and a SOS-style semantics for the new construct.

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

Turn over/...

3 The abstract syntax for the language While is given by:

$$
\begin{array}{lcl}
x & \in & \text{Variable} \\
a & \in & \text{Arithmetic} - \text{expression} \\
b & \in & \text{Boolean} - \text{expression} \\
S & \in & \text{Statement} \\
S & ::= & x := a \mid S1;S2 \mid \texttt{if } b \texttt{ then } S1 \texttt{ else } S2 \mid \texttt{skip} \mid \texttt{while } b \texttt{ do } S
\end{array}
$$

The syntax of expressions (both arithmetic and boolean) is unspecified.

a Briefly describe each of the three main approaches to specifying programming language semantics: natural, structural operational, and denotational semantics. Identify their comparative strengths and weaknesses.

b Assuming that:

$$\text{if } \langle S_1, s_1 \rangle \Rightarrow^* s_2 \text{ then } \langle S_1; S_2, s_1 \rangle \Rightarrow^* \langle S_2, s_2 \rangle,$$

show that, for the language While, structural operational and natural semantics coincide. The proof will follow an inductive reasoning; it suffices to show a base case and one (non-trivial) inductive step in each direction.

c The While language is extended with parameterless procedures. In particular, there is a new calling statement:

$$S ::= \ldots \mid \texttt{call } p$$

i) Assuming that the language has dynamic scope rules, write down the Natural Semantics for the `call` statement.

ii) Assuming that the language has static scoping for procedures (but not variables), write down two alternative Natural Semantics for the `call` statement.

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

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

$$
\begin{array}{lcl}
x & \in & \text{Variable} \\
a & \in & \text{Arithmetic} - \text{expression} \\
b & \in & \text{Boolean} - \text{expression} \\
S & \in & \text{Statement} \\
S & ::= & x := a \mid S1;S2 \mid \texttt{if}\ b\ \texttt{then}\ S1\ \texttt{else}\ S2 \mid \texttt{skip} \mid \texttt{repeat}\ S\ \texttt{until}\ b
\end{array}
$$

The syntax of expressions (both arithmetic and boolean) is unspecified.

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

b Let $f : D \to D$ be a continuous function on the ccpo $(D, \sqsubseteq)$ and let $d \in D$ satisfy

$$fd \sqsubseteq d.$$

Show that FIX f approximates d.

c Use the denotational semantics to compute the meaning of:

$$\texttt{repeat}\ x := x + 1\ \texttt{until}\ x = 10$$

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

End of Paper