UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2002

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 City and Guilds of London Institute
This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science

PAPER C242=MC242

OPERATIONAL SEMANTICS

Friday 3 May 2002, 14:30
Duration: 90 minutes
(Reading time 5 minutes)

Answer THREE questions

Paper contains 4 questions
Calculators not required

1) The following is the abstract syntax of a *Mouse control language*:

$$
\begin{aligned}
a &\in \textit{Arithmetic Expressions}\\
p &\in \textit{Program}\\
p &::= \texttt{north} \mid \texttt{east} \mid \texttt{south} \mid \texttt{west} \mid \texttt{run}(n) \mid \texttt{rest}(n) \mid p_1 \texttt{;} p_2
\end{aligned}
$$

A program written in this language controls a mechanical Mouse that moves around on an infinite board, composed of fields, that are either occupied (by, for example, walls or other mice) or are accessible.

- Using **north**, **east**, **south**, and **west**, the Mouse changes direction.
- Using **run** (n), it will run n fields from its current position in the direction that it is pointing. If, during this run, the Mouse encounters an occupied field, it will stop, and randomly choose to change direction.
- Using **rest**(n), the Mouse will stay in the same location for n 'ticks'.

Many mice can move around on the board simultaneously, but you can assume that they are controlled by *one* CPU (so synchronisation is not an issue).

The state of the Mouse is represented by a triple: $\langle pos, d, \mathtt{F}\rangle$. The first element *pos* gives the Mouse's current location in the field, in Cartesian co-ordinates $\langle x, y\rangle$; the second element d gives the direction the Mouse is heading (either **NORTH**, **EAST**, **SOUTH**, or **WEST**); the third element **F** is an (infinite) double array of booleans indicating whether a field on the board is occupied or not.

i. Assuming the existence of the function $n \mapsto \mathcal{N}[\![n]\!]$ that defines the semantics of numbers, give the *Natural Semantics* of Mouse programs. You can use the following abbreviation:

$$
\begin{aligned}
\langle x, y\rangle + d \cdot i &= \langle x, y{+}i\rangle, \text{ if } d = \textbf{NORTH},\\
&= \langle x{+}i, y\rangle, \text{ if } d = \textbf{EAST},\\
&= \langle x, y{-}i\rangle, \text{ if } d = \textbf{SOUTH}, \text{ and}\\
&= \langle x{-}i, y\rangle, \text{ if } d = \textbf{WEST}
\end{aligned}
$$

ii. Is the semantics of the previous part *deterministic*?

Are Mouse programs *terminating*?

Motivate your answer in both cases.

iii. An abstract *Mouse Machine* itself has configurations

$$\langle c, e, s\rangle \in \textit{Code} \times \textit{Stack} \times \textit{State},$$

2) The abstract syntax for the language **from-to-loop** is given by:

$$
\begin{array}{rcl}
x & \in & \textit{Variables} \\
a & \in & \textit{Arithmetic Expressions} \\
b & \in & \textit{Boolean Expressions} \\
S & \in & \textit{Statements}
\end{array}
$$

$$
\begin{array}{rcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\
b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \;\&\; b_2 \\
S & ::= & x := a \mid S_1 \texttt{;} S_2 \mid \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid \\
 & & \texttt{skip} \mid \texttt{from}\ a_1\ \texttt{to}\ a_2\ \texttt{do}\ S
\end{array}
$$

The intention of the '**from** a_1 **to** a_2 **do** S' is that, first, the values of a_1 and a_2 will be checked. If a_2 is greater than or equal to a_1, then S will executed. After that, the entry test will be repeated, checking now the values of a_2 and $a_1 + 1$. This repeats until the test fails.

i. Assume the existence of the functions $a \mapsto \mathcal{A}\,[\![a]\!]\,s$ and $b \mapsto \mathcal{B}\,[\![b]\!]\,s$ that define the semantics of arithmetic and boolean expressions. Define the Structural Operational Semantics for **from-to-loop**.

ii. What does it mean for two programs to be *semantically equivalent*? Show that

$$\texttt{from}\ a_1\ \texttt{to}\ a_2\ \texttt{do}\ S$$

and

$$\texttt{if}\ (a_1 \leq a_2)\ \texttt{then}\ S; \texttt{from}\ (a_1 + 1)\ \texttt{to}\ a_2\ \texttt{do}\ S\ \texttt{else skip}$$

are semantically equivalent.

iii. Let (part of) an abstract machine be defined by:

$$
\begin{array}{rcl}
\texttt{inst} & ::= & \texttt{PUSH}{-}n \mid \texttt{ADD} \mid \texttt{MULT} \mid \texttt{SUB} \mid \texttt{TRUE} \mid \texttt{FALSE} \mid \texttt{EQ} \mid \texttt{LE} \mid \texttt{AND} \mid \texttt{NEG} \mid \\
 & & \texttt{FETCH}{-}x \mid \texttt{STORE}{-}x \mid \texttt{NOOP} \mid \texttt{BRANCH}(c, c) \\
c & ::= & \epsilon \mid \texttt{inst} : c
\end{array}
$$

where its operational semantics '$\cdot \rhd \cdot$' is defined as in the notes.

Assuming the definition of $\mathcal{CA}\,[\![a]\!]$ and $\mathcal{CB}\,[\![b]\!]$ that, respectively, define the translation of *Arithmetic Expressions* and *Boolean Expressions* to *Code*, give the definition for $\mathcal{CS}\,[\![S]\!]$. Notice that this implies that you will have to extend also the definition of **inst**.

The three parts carry 35%, 30%, and 35% of the marks, respectively.

4) i. Consider the space of partial functions from *State* to *State*.

A) Give the definition of a *partial ordered* set.
B) How is the partial order relation $\sqsubseteq$ defined on functions?
C) Give the definition of a *least upper bound.*
D) Give the definition of a *chain.*
E) Give the definition of a *ccpo.*
F) When is a function *monotone*?
G) What is a *continuous* function?

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

$$
\begin{aligned}
x &\in \textit{Variables} \\
a &\in \textit{Arithmetic Expressions} \\
b &\in \textit{Boolean Expressions} \\
S &\in \textit{Statements}
\end{aligned}
$$

$$
\begin{aligned}
a &::= n \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\
b &::= \texttt{true} \mid \texttt{false} \mid (a_1 = a_2) \mid (a_1 \leq a_2) \mid (\neg b) \mid (b_1 \;\&\; b_2) \\
S &::= x := a \mid S_1 \texttt{;} S_2 \mid \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid \texttt{skip} \\
&\quad \mid \texttt{loop}\ S_1\ \texttt{exif}\ b\ S_2\ \texttt{endl}
\end{aligned}
$$

(The idea of the `loop` $-$ `endl` construct is that, in contrast to a `while` loop, a number of statements will be executed before testing the boolean. If the boolean tests true, then execution of the loop is terminated.)

Assuming the existence of suitable functions $a \mapsto \mathcal{A}[\![a]\!]\, s$ and $b \mapsto \mathcal{B}[\![b]\!]\, s$ for the semantics of both arithmetic and boolean expressions, give the *Denotational semantics* of statements.

iii. *Calculate* the denotational semantics for

$$\texttt{loop skip exif}\ (x < 0)\ (x := x-1)\ \texttt{endl}$$

The three parts carry, respectively, 35%, 30%, and 35% of the marks.

End of Paper