UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

MEng Honours Degrees in Computing Part IV
MSc Degree in Foundations of Advanced Information Technology
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

PAPER 4.92

THEORIES OF SPECIFICATION AND VERIFICATION

Tuesday, May 7th 1996, 10.00 - 12.00

Answer THREE questions

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

1 This question concerns a robot control system, with components:

- A press, which responds to signals to close and open, and whose position can be determined;
- A robot (with two arms), which can rotate clockwise or anti-clockwise.

The robot picks up metal pieces from a table, rotates until it is positioned at the press, picks up a forged piece from the press, and then rotates until it is positioned at a deposit belt, where it deposits the forged piece. Finally it rotates to the press, deposits the unforged piece in the press and continues back to the table to complete the cycle.

Figure 1 shows the physical layout of the robot.

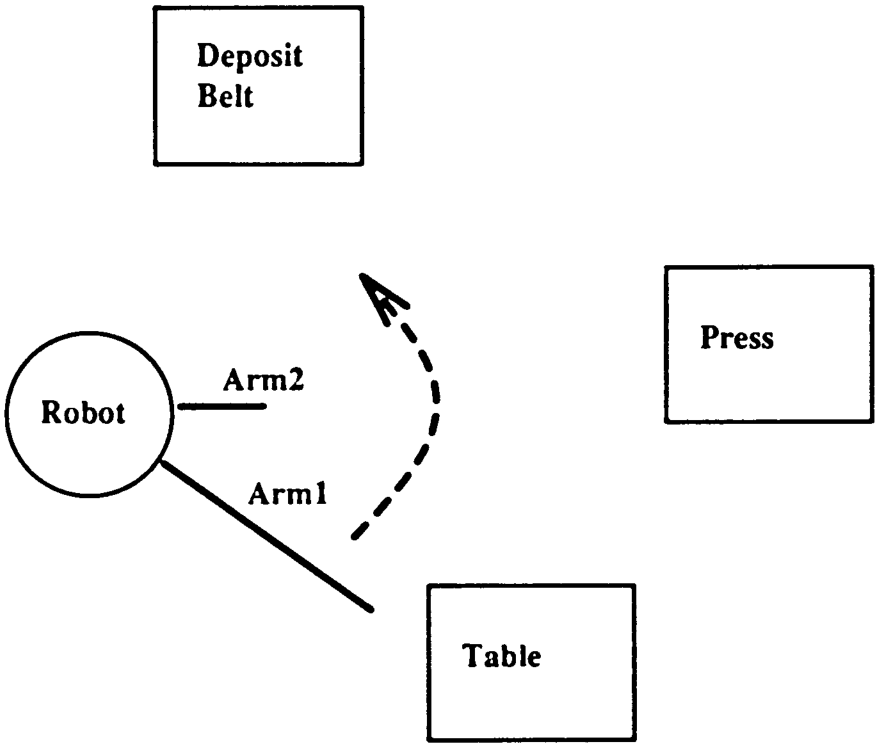

Figure 1: Robot System

There are therefore four significant states for the robot: (i) positioned at the table; (ii) positioned at the press for pickup; (iii) positioned at the deposit belt; (iv) positioned at the press for deposit. Operations are needed to carry out the pickup and deposit steps corresponding to each of these positions, and other operations are needed to change the state from one position to the next.

a Formalise the robot as a machine with appropriate state variables and operations (**complete_pickup_from_table**, **rotate_table_press**, **complete_deposit_at_press**, etc. – list at least 4 of the 8 operations) to perform the robot actions and to move between the above 4 states. Include relevant operation preconditions.

b Enhance this machine with a variable **rotation** $\in$ ANGLE which represents the current rotational position of the robot. The table position is at angle 0, the pickup from press position is a constant **press1_angle** $\in$ ANGLE, the deposit belt position is ...*PTO*

deposit_angle and the press deposit position is **press2_angle**. Modify the operation preconditions and effects to take account of this new variable. Use a suitable inclusion mechanism to access the **PCTypes** machine below to give you access to the **ANGLE** type and these constants.

```
MACHINE PCTypes
SETS
  ANGLE
CONSTANTS
  table_angle, press1_angle, deposit_angle, press2_angle
PROPERTIES
  table_angle = 0 & press1_angle : ANGLE &
  deposit_angle : ANGLE & press2_angle : ANGLE &
  ANGLE = 0..359
END
```

The two parts carry, respectively, 60% and 40% of the marks.

2

a Define the data and initialisation parts of a machine to represent the concept of a "node" as defined in Figure 2. A node has a current state (either "idle", "constructing" or "constructed"), a parent node. a set of "children" nodes, and a set of "neighbour" nodes. It also has a numeric identifier.

b Define an operation to create a node with an initially idle state, and empty sets of children and neighbour nodes, and a given parent node (which can be itself, in the case of the root node of a tree, for example) and identifier.

c Define an operation to move a given node from the idle to the constructing state.

d Define an operation to add a node to the neighbour set of a given node.

e Define an operation to return the number of nodes currently in the "constructing" state.

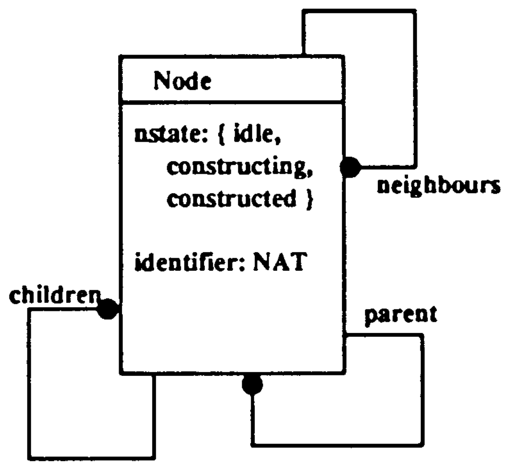

Figure 2: Object Model of Node

The five parts carry, respectively, 20%, 20%, 20%, 20% and 20% of the marks. ... *PTO*

3 Consider the specifications below of a consumer object and a producer object.

PRODUCER

<u>data sorts</u>: ITEM, BOOL

<u>data functions</u>: boolean operations

<u>attribute symbols</u>:
  current : ITEM
  waiting : BOOL

<u>action symbols</u>
  produce(ITEM)
  store(ITEM)

<u>axioms</u> i : ITEM

p0 Usual axioms for booleans
p1 produce(i) $\rightarrow$ Xcurrent = i
p2 produce(i) $\rightarrow$ Xwaiting = true
p3 store(i) $\rightarrow$ Xwaiting = false
p4 produce(i) $\rightarrow$ waiting = false
p5 store(i) $\rightarrow$ current = i
p6 store(i) $\rightarrow$ waiting = true
p7 waiting = true $\rightarrow$ Fstore(current)

CONSUMER

<u>data sorts</u>: ITEM, BOOL

<u>data functions</u>: boolean operations

<u>attribute symbols</u>:
  current : ITEM
  waiting : BOOL

<u>action symbols</u>
  consume(ITEM)
  extract(ITEM)

<u>axioms</u> i : ITEM

p0 Usual axioms for booleans
p1 extract(i) $\rightarrow$ Xcurrent = i
p2 extract(i) $\rightarrow$ Xwaiting = true
p3 consume(i) $\rightarrow$ Xwaiting = false
p4 extract(i) $\rightarrow$ waiting = false
p5 consume(i) $\rightarrow$ current = i
p6 consume(i) $\rightarrow$ waiting = true

... PTO

p7 waiting = true → Fconsume(current)

a Specify in the same formalism a buffer of items which behaves as a bag of items. That is, the buffer records the number of times a particular item has been *put* in the buffer less the number of times it has been extracted (via the action *get*). (hence the buffer does <u>not</u> support the FIFO principle.)

b Build a system in which a single producer simultaneously stores an item in 2 independent buffers.

c Chain two producers and two buffers so that the first producer stores items in the first buffer and the second producer independently picks up items from this first buffer which are then stored in the second buffer. (Do not alter in any way the specification of the producer to do this.)

d Modify the specification of the consumer so that it has two independent extract actions (one for each of the two buffers to which the consumer may be attached). The only rule that must be observed is that the two extracts are mutually exclusive and that an item once extracted must be consumed before another instance of the same or the other extract can be invoked. Attach this new consumer to two buffers so that the consumer can extract from them (via the two different extract actions).

e Using the original version of the consumer, build a system in which it is attached to two buffers so that successive calls of *extract* are alternately applied to the two different buffers. (Hint: you must build an interface to the two buffers which will cause this successive alternation to happen.)

The five parts carry, respectively, 20%, 20%, 15%, 15% and 30% of the marks.

4

a Given a language Θ define the concept of a Θ *structure*. When is a Θ structure a *locus*? Give the *locality axiom* for Θ and explain its intended meaning.

b Explain what is meant by *safety* and *liveness* properties of objects. Give an example of each kind of property from the example specifications of a consumer and a producer in question 3 above.

c Explain the motivation for the 'open semantics' used for objects. Explain how the locality axiom can be used to justify an inference rule which supports the use of case-based inductive reasoning about safety properties of objects.

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

End of paper.