UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2004

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
MSc in Computing for Industry
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 C223=MC223

CONCURRENT PROGRAMMING

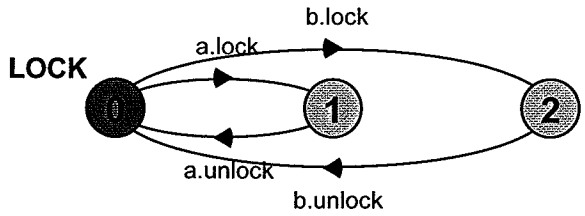
Thursday 6 May 2004, 14:30
Duration: 120 minutes

Answer THREE questions

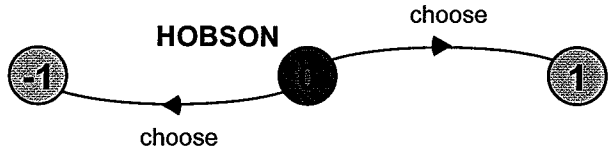
Paper contains 4 questions
Calculators not required

1a For each of the following Labelled Transition Systems (LTS), give an equivalent FSP specification.

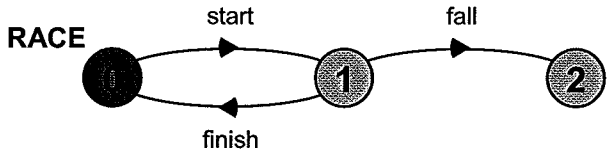
i)



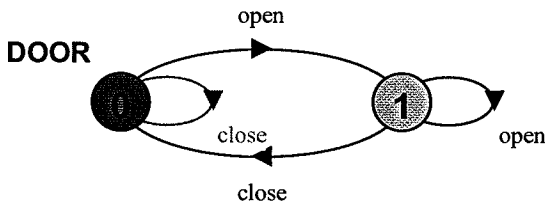
ii)



iii)



iv)



b. For each of the following FSP specifications, give an equivalent LTS.

i) `SQUARE = (in[i:1..2]->out[i*i]->SQUARE).`

ii)
```
CLOCK          = CLOCK[0],
CLOCK[i:0..4] = (when(i<4) tick->CLOCK[i+1]).
```

iii)
```
minimal
DICE = (throw[i:1..6]->(when (i==6)again->DICE)).
```

iv)
```
ALICE = (start->alice.finish->ALICE).
BOB   = (start->bob.finish->BOB).
||AB  = (ALICE||BOB).   //draw LTS for AB only
```

c. If `P` is a deterministic process, then `P||P = P`. Explain why this is not the case for a non-deterministic process. Illustrate your answer using an example.

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

2a What is the *alphabet* of a process specified in FSP?

b A timer service is required that cyclically counts from 0 to 59 seconds and then returns to zero again. The timer increments its count every time it is signalled by an external **tick** action. Processes using the timer service can request to be executed when the timer reaches a particular count using the action **at[t]** - this delays the invoking process until the counter within the timer service has the value **t**. Give an FSP model **TIMER** for the timer service using the following definitions:

```
const Max = 60
range Time = 0..Max-1
```

c The definition of a system model which includes processes that use the **TIMER** service is given below:

```
PROCESS(I=2) = (at[I] -> run -> STOP)+ {at[Time]}.

||SYS = ({a,b}::TIMER || a:PROCESS(2) || b:PROCESS(7))
        /{tick/{a,b}.tick}.
```

i) Give a trace of this system that includes ten **tick** actions.

ii) Explain clearly why the alphabet extension is required in the definition of **PROCESS**.

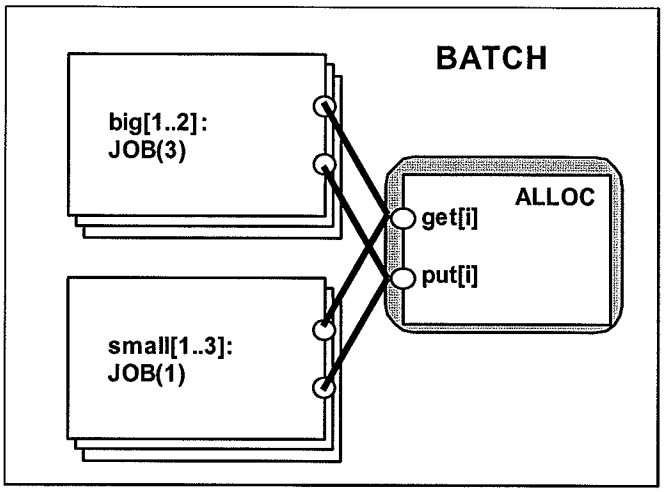
iii) Give a sample trace of the system if the alphabet extension is excluded; the trace should be different from that given as an answer to part i).

d Give an implementation as a Java class of the timer service (part b) that you modelled in FSP as the process **TIMER**.

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

3a Explain what is meant by safety and liveness properties with respect to concurrent programs.

b The overall model structure of a batch processing system is shown below:



A job entering the system must acquire memory blocks from an allocator before it can run and returns the blocks when it has finished running. A job is blocked until sufficient blocks become available. There are two types of jobs – big jobs require three blocks and small jobs require only one. Given models for a job and allocator :

```
const NB      = 4 //maximum number of allocatable blocks
set All       =  {{get,put}[1..NB]}

JOB(B=1)      = (get[B]->run->put[B]->JOB)+ All.

ALLOC               = ALLOC[NB],
ALLOC[i:0..NB]      = (when (i>0)get[j:1..i] -> ALLOC[i-j]
                      |put[j:1..NB]          -> ALLOC[i+j]
                      ).
```

i) Specify the composite process **BATCH** in FSP.

ii) Specify two properties, one for small jobs and one for big jobs that assert that jobs eventually get allocated memory blocks.

iii) Specify a **BATCH** system with adverse scheduling conditions to model a heavily loaded system.

iv) Briefly outline a scenario in which one of the properties you have specified in ii) might be violated in the system you have specified in iii).

c Implement the specifications for each of the entities (**JOB, ALLOC**) in Java. Include the definition of a method **void build(int NB)** which creates the objects required for **BATCH**.

The three parts carry, respectively, 10%, 40%, 50% of the marks

4a Explain how *safety property* automata in FSP can observe the correct execution of a model without constraining it.

Draw the Labelled Transition System for the following safety property:

```
property
TIMELY=(arrive      ->TIMELY | begin_lecture->LATE),
LATE  =(end_lecture->TIMELY | begin_lecture->LATE).
```

and give *two* examples of traces that violate the property.

b A reservoir has a maximum capacity of `N` cubic metres of water. The action `fill[x]` fills the reservoir with `x` cubic metres where `x` is in the range `T = 1..N`. Similarly the action `empty[x]` empties the reservoir of `x` cubic metres where `x` is in the range `T`. Specify a safety property that asserts that a model of the reservoir does not overflow or underflow.
(*NB: Concise models will get more marks*)

c The following models `N` processes synchronizing on the action `sync` in that all of them must execute the sync action before they can continue.

```
const N = 3
P = (before ->sync->after->P).
||SYS = (p[1..N]:P)/{sync/p[1..N].sync}.
```

Write a Java class with a method `sync()` that implements synchronization as modeled above.

d It is possible for the following system to deadlock. Explain how this deadlock occurs and relate it to one of the four necessary and sufficient conditions for deadlock to occur.

```
Alice = (call.bob -> wait.chris -> Alice).
Bob   = (call.chris -> wait.alice -> Bob).
Chris = (call.alice -> wait.bob -> Chris).

||S = (Alice || Bob || Chris) /{call/wait}.
```

The following model attempts to fix the problem by allowing Alice, Bob and Chris to timeout from a call attempt. Is a deadlock still possible? If so describe how the deadlock can occur and give an execution trace leading to the deadlock.

```
Alice = (call.bob -> wait.chris -> Alice
        | timeout.alice -> wait.chris ->Alice).
Bob   = (call.chris -> wait.alice -> Bob
        | timeout.bob -> wait.alice ->Bob).
Chris = (call.alice -> wait.bob -> Chris
        | timeout.chris -> wait.bob ->Chris).

 ||S = (Alice || Bob || Chris) /{call/wait}.
```

The four parts carry, respectively, 30%, 20%, 30%, 20% of the marks