Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Concurrency and Process Algebra

Date: Thursday 28th May 2015
Time: 09:45 - 11:45

Please answer any THREE Questions from the FIVE Questions provided

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
1. Modelling and implementing concurrent systems.

a) In the context of concurrent systems, explain briefly but clearly what is meant by the following:

i. Mutual exclusion,

ii. A monitor, and the concepts of active and passive processes.

Explain the mechanisms that a monitor may provide to allow controlled access by multiple processes. (3 marks)

b) A shop has an entrance door and a separate exit door. The shopkeeper may open the shop (if it is closed) and then close it. When it is opened customers are allowed to enter and leave. When the shop is closed, customers may leave, but not enter. We model these elements in the process algebra FSP as follows.

\[
\text{ENTRANCE} = \text{enter} \rightarrow \text{ENTRANCE}.
\]
\[
\text{EXIT} = \text{leave} \rightarrow \text{EXIT}.
\]
\[
\text{KEEPER} = \text{open} \rightarrow \text{close} \rightarrow \text{KEEPER}.
\]

A control system controls the doors and allows customers to enter and leave up to a maximum capacity for the shop, \( \text{MAX} \).

Model the control system as an FSP process \( \text{CONTROL} \). The control may be in one of two states, \( \text{OPENED} \) or \( \text{CLOSED} \), which may each be modelled as a process indexed by the number of customers in the shop and with guards on the actions. (5 marks)

c) The shop may then be described as the following composite process.

\[
\text{SHOP} = (\text{ENTRANCE} || \text{EXIT} || \text{KEEPER} || \text{CONTROL}).
\]

Explain how synchronisation in FSP ensures the correct behaviour of the shop. Give an example of a correct, and an incorrect, trace of actions. (2 marks)

d) We now wish to implement the \( \text{CONTROL} \) process. Give an implementation as a class definition in Java with methods for entering and leaving the shop, and opening and closing the shop. You may use a boolean variable to implement the two states of the \( \text{CONTROL} \) process. Ensure that you handle the synchronisation and use of locks correctly. You need not implement the other processes above. (7 marks)

e) Explain briefly (i) how the model and its implementation relate to the concept of a monitor, and (ii) how process interaction in the FSP model is captured in the Java implementation. (3 marks)
2. On FSP rules and derivations.

a) Give the inference rules for the parallel operator $||$ in FSP. Explain how synchronised and unsynchronised transitions of concurrent systems are described by these rules. (3 marks)

Consider the following FSP process definitions which provide a simple model of a petrol station with one pump and a customer, and the requirement that fuel needs to be prepaid.

CUSTOMER = ( pay $\rightarrow$ hoseready $\rightarrow$ fuel $\rightarrow$ replacehose $\rightarrow$ CUSTOMER $\mid$ hoseready $\rightarrow$ replacehose $\rightarrow$ CUSTOMER )

PUMP = ( pumpon $\rightarrow$ ONPUMP $\mid$ hoseready $\rightarrow$ replacehose $\rightarrow$ PUMP ),

ONPUMP = ( hoseready $\rightarrow$ fuel $\rightarrow$ replacehose $\rightarrow$ PUMP ).

CASHIER = ( pay $\rightarrow$ pumpon $\rightarrow$ CASHIER $\mid$ hoseready $\rightarrow$ replacehose $\rightarrow$ CASHIER ).

$||$PETROLSTATION = ( CASHIER $||$ CUSTOMER $||$ PUMP ).

b) Using transition rules for FSP, provide a detailed derivation of the pay transition that the PETROLSTATION process can make. Carefully explain all the rules that you use at each step of the derivation. (8 marks)

c) Explain clearly what happens to the PETROLSTATION, in terms of transitions and rules, when the CUSTOMER attempts to get fuel without paying by starting with the hoseready action instead of the pay action. (3 marks)

d) Describe precisely how a labelled transition system may be constructed from an FSP process definition using FSP rules. (3 marks)

e) Now draw a labelled transition system that corresponds to the composite FSP process PETROLSTATION above. You should construct a system with the minimum number of states. (3 marks)
3. On the equivalence of FSP processes.

a) Give the definition of \textit{(not an algorithm for)} strong bisimilarity between two labelled transition systems, and hence define strong bisimulation for FSP processes. \hfill (3 marks)

b) By giving two example processes, show that processes may have the same set of traces but not be strongly bisimilar. Justify your answer. \hfill (2 marks)

c) Describe an algorithm for computing whether two FSP processes are strongly bisimilar. Your description should clearly explain all the steps of the algorithm. \hfill (6 marks)

d) Consider the following FSP processes \( P \) and \( X \).

\[
P = ( a \rightarrow Q \mid a \rightarrow R \mid d \rightarrow S ),
\]
\[
Q = ( a \rightarrow R \mid b \rightarrow P ),
\]
\[
R = ( a \rightarrow R \mid b \rightarrow P ).
\]

\[
X = ( a \rightarrow Y \mid d \rightarrow W ),
\]
\[
Y = ( a \rightarrow Y \mid b \rightarrow Z ),
\]
\[
Z = ( a \rightarrow Y \mid d \rightarrow W ).
\]

Use the algorithm you described in answer to Part (3c) above to determine whether or not process \( P \) is strongly bisimilar to process \( X \). Show your working - the steps of the algorithm run on this example. \hfill (6 marks)

e) What is the minimum labelled transition system that is strongly bisimilar to the labelled transition system for \( P \)? Justify your answer using the results of your algorithm, or otherwise. \hfill (3 marks)
4. On properties and property checking.

a) Explain what is meant by a safety property and a liveness property for a concurrent system. Give an example of a real system together with a safety property and a liveness property that are required to hold for the system. (4 marks)

b) A safety property is defined in FSP as a process definition prefixed by the keyword property, as in the example below:

```
property LIGHTS_PROPERTY =
    ( red -> amber -> green -> LIGHTS_PROPERTY ).
```

Explain the difference between LIGHTS_PROPERTY and the process LIGHTS defined below. In particular, draw the labelled transition systems for both and explain how they are constructed. (3 marks)

```
LIGHTS = ( red -> amber -> green -> LIGHTS ).
```

c) Consider the following FSP model of a controller for a traffic light system.

```
range S = 0..3

SIGNAL = SIGNAL[0],
SIGNAL[s: S] =
    ( when (s==0) red -> SIGNAL[1]
    | when (s==1) amber -> SIGNAL[2]
    | when (s==2) green -> SIGNAL[3]
    | when (s==3) red -> SIGNAL[0] ).
```

Draw the labelled transition system corresponding to the SIGNAL process. (3 marks)

d) Now draw the labelled transition system corresponding to the following composite process. (5 marks)

```
SIGNAL || LIGHTS_PROPERTY
```

e) Using this labelled transition system, show that the SIGNAL process does not satisfy the LIGHTS_PROPERTY. Justify your answer with a trace from the start state. (2 marks)

f) Correct the definition of the SIGNAL process so that the desired safety property is satisfied. (3 marks)
5. Concurrency concepts.

a) Explain briefly but clearly the following concepts applied to concurrent systems. You should use illustrative examples. (4 marks)
   
i) Deadlock
   ii) Livelock

b) What four conditions on a system ensure that deadlock is a possibility? Give a clear explanation of each condition. (4 marks)

c) Consider the following FSP description of a simple case of the dining philosophers problem with five (N=5) philosophers sitting around a circular table, each with a shared right and left fork. A philosopher can eat only if holding both a right and a left fork:
   
   \[
   \text{FORK} = (\text{get} \rightarrow \text{put} \rightarrow \text{FORK}).
   \]

   \[
   \text{PHIL} = (\text{sitdown} \rightarrow \text{right.get} \rightarrow \text{left.get} \\
   \rightarrow \text{eat} \\
   \rightarrow \text{left.put} \rightarrow \text{right.put} \\
   \rightarrow \text{arise} \rightarrow \text{PHIL}).
   \]

   \[
   ||\text{DINERS(N=5)} = \\
   \forall [i:0..N-1] \\
   ( \text{phil[i]}:\text{PHIL} || \\
   \{ \text{phil[i].left}, \text{phil[(((i-1)+N) mod N].right} \}::\text{FORK} ).
   \]

Here \( \text{mod N} \) is the remainder after division by \( N \), and the \( \forall \) construct gives the iterated parallel composition, i.e.

\[
\forall [i:0..N-1] \ P[i] = (P[0] || ... || P[N-1]).
\]

i) Give an example of a trace to deadlock and explain the deadlock that occurs. (2 marks)

ii) Explain clearly how each of the conditions you described in answer to Part (b) above applies to this model. (2 marks)

iii) For each of these conditions, describe a solution, i.e. describe FOUR separate modifications of the above system, each of which prevents one of the conditions holding. Your descriptions should be clearly explained in terms of the philosophers, the table and access to the forks, but need not be expressed in FSP. (4 marks)

iv) Express ONE of these solutions in FSP by modifying the FSP model above. (4 marks)