Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Concurrency and Process Algebra

Date: Wednesday 5th June 2013
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

[PTO]
1. Modelling concurrent systems in FSP.

We consider a very simple warehouse model in FSP. In this simplified model the warehouse stores only one type of item. A factory produces the items which then get stored one-by-one in the warehouse. Delivery vans can take an item from the warehouse and deliver it to a consumer. In FSP, we model the factory and consumer as the simple processes:

FACTORY = (store -> FACTORY).
CONSUMER = (deliver -> CONSUMER).

The warehouse has a fixed capacity of 100 say. Model the warehouse as an FSP process. Hint: You will need to use an indexed process definition to record the number of items in the warehouse. (4 marks)

Define the composite system of a factory, a warehouse and a consumer, and explain clearly how the interaction of the processes takes place. (3 marks)

In concurrency, explain briefly what is meant by a monitor and describe the main features of monitors. How do monitors relate to your FSP model? (4 marks)

Give an implementation of the warehouse, factory and consumer in Java. You may describe the warehouse as an array. You should define the required three classes, making clear which objects are threads and implementing the actions as methods, ensuring that their interaction is properly handled. (9 marks)
2. On FSP rules.

Consider the following FSP process definitions which provide a very simple model of a museum, with an east door that allows visitors to enter, a west door allowing visitors to leave, and a director that can open or close the museum. When the museum is closed, visitors may leave but not enter.

\begin{verbatim}
const N = 5

EAST = (arrive -> EAST).
WEST = (leave -> WEST).
DIRECTOR = (open -> close -> DIRECTOR).

CONTROL = ( open -> OPENED
| leave -> CONTROL ),
OPENED = ( close -> CONTROL
| arrive -> OPENED
| leave -> OPENED ).

||MUSEUM = ((EAST || WEST) || (DIRECTOR || CONTROL)).
\end{verbatim}

a) Using transition rules for FSP, provide a detailed derivation of the open transition that the CONTROL process can make. Carefully explain the rules that you use.

(4 marks)

b) Write down the three transition rules for the parallel composition of two FSP processes. Hence construct a detailed derivation of the open transition that the composite process MUSEUM can make.

(6 marks)

c) Now formally show, using the FSP transition rules, that the process MUSEUM can perform a close transition immediately after this open transition.

(5 marks)

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

(2 marks)

e) Use this analysis to draw a labelled transition system that corresponds to the composite FSP process MUSEUM. You should try to construct a system with the minimum number of states.

(3 marks)
3. On the equivalence of FSP processes.

a) Provide and explain a formal definition of (not an algorithm for) strong bisimilarity between two FSP processes. (4 marks)

b) By defining two example processes, show that two processes may have the same set of traces but not be strongly bisimilar. Justify your answer. Give a third FSP process that can be used to distinguish your two processes, i.e. when the third process is combined with the first two by parallel composition, different sets of traces result. (4 marks)

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

d) Given the following FSP definitions

\[ P = ( a \rightarrow ( b \rightarrow Q ) | c \rightarrow Q ), \]
\[ Q = ( c \rightarrow P | a \rightarrow b \rightarrow P ). \]
\[ X = (a \rightarrow Y | c \rightarrow X ), \]
\[ Y = (b \rightarrow X ). \]

use the algorithm you gave in answer to part 3c above to determine whether or not the process \( P \) is strongly bisimilar to \( X \). (4 marks)

e) What is the minimal FSP process that is strongly bisimilar to \( P \)? Justify your answer. (2 marks)
4. On properties and property checking.

a) Explain what is meant by the terms safety property and liveness property, giving one example of each. (4 marks)

b) How are safety properties expressed in FSP? In particular, in terms of labelled transition systems, what is the difference between the property process P and the (normal) process Q below?

\[
\text{property } P = (\text{in } \rightarrow \text{ out } \rightarrow P).
\]
\[
Q = (\text{in } \rightarrow \text{ out } \rightarrow Q).
\]

(4 marks)

c) Given the following FSP

\[
\text{Sender} = (\text{in } \rightarrow \text{ snd } \rightarrow \text{ ack } \rightarrow \text{ Sender}).
\]
\[
\text{Receiver} = (\text{snd } \rightarrow \text{ ack } \rightarrow \text{ out } \rightarrow \text{ Receiver}).
\]
\[
||\text{Composition} = (\text{Sender} || \text{Receiver})\backslash\{\text{snd, ack}\}.
\]

Write down and justify what FSP process, let it be named Check, should be constructed in order to check whether the property P, given above in Q4b, holds for the Composition process. (2 marks)

d) Draw the labelled transition systems corresponding to the Composition process, to the property process P and your Check process. (6 marks)

e) Does the safety property P hold for the Composition process? If not, use the information so gained to identify the fault and fix the problem in one of the above definitions (you may assume that the property P is indeed correctly specified). (4 marks)
5. Concurrency concepts.

a) Explain briefly but clearly the following concepts applied to concurrent systems. You should use illustrative examples. (5 marks)

i) Deadlock
ii) Livelock

b) What four conditions on a system ensure that deadlock is a possibility? (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:

\[ \text{FORK} = (\text{get} \rightarrow \text{put} \rightarrow \text{FORK}). \]

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

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

Here the forall construct gives the iterated parallel composition, i.e.

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

i) Explain clearly how this system can reach deadlock and give a trace to deadlock. (2 marks)

ii) Modify the system so that deadlock is impossible. You may consider one of the following solutions or one of your own invention: (6 marks)

- Make the picking up of a philosopher’s two forks an atomic step, for example by modifying the philosopher process above,
- Introduce a ‘butler’ process which imposes a scheduling on the philosophers. One possible scheduling is to allow only N-1 philosophers to sit at the table at any one time.

iii) Explain clearly why deadlock is not possible in your modified system. (3 marks)