Concurrency and Process Algebra

Date: Thursday 19th May 2011
Time: 14:00 - 16:00

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 concurrent systems in FSP.

a) An FSP model of a burglar alarm system is under development. A first attempt at modelling the behaviour of the control unit is given below by the FSP process **Control**.

hint: The **Control** process is defined as follows:

```fsharp
let Control = ( setCode[x:Code] -> UnArmed[x] ),

UnArmed[x:Code] =
    ( displayCode[x] -> UnArmed[x]
    | setCode[y:Code] -> UnArmed[y]
    | on -> Armed[x] ),

Armed[x:Code] =
    ( enterCode[x] -> UnArmed[x]
    | trigger -> Triggered[x] ),

Triggered[x:Code] =
    ( enterCode[x] -> Armed[x]
    | soundAlarm -> Triggered[x] ).
```

Once the alarm system is in the **Armed** state of behaviour, entry of the correct code via **enterCode[x]** action will disable the alarm system. Unfortunately, this model does not accept incorrect codes. Explain why this is so. (2 marks)

b) Now modify the code entry mechanism of the **Control** process (to become **Control2**) to allow for any code to be entered as follows. Correct codes will be acknowledged by a **beep** event followed by the control system becoming disabled, i.e. moving into the **UnArmed** state. Incorrect codes will result with a **codeError** action and the control system remains in the **Armed** state of behaviour. (4 marks)

```
let Control2 = ( setCode[x:Code] -> UnArmed[x] ),

UnArmed[x:Code] =
    ( displayCode[x] -> UnArmed[x]
    | setCode[y:Code] -> UnArmed[y]
    | on -> Armed[x] ),

Armed[x:Code] =
    ( enterCode[x] -> UnArmed[x]
    | trigger -> Triggered[x] ),

Triggered[x:Code] =
    ( enterCode[x] -> Armed[x]
    | soundAlarm -> Triggered[x] ).
```

The sensor is enabled via an **on** action, detected movement then causes a **trigger** action to occur and the sensor moves to a **SensorTriggered** state of behaviour. A triggered sensor can be reset to detect movement via the **reset** action, and the **off**
action disables the sensor. Clearly, the actions of the sensor need to be shared with your Control2 process. The parallel process BurglarAlarm then represents the combined behaviour of the sensor and the control system.

$$\text{||BurglarAlarm} = \{ \text{Control2 || Sensor} \}.$$  

The above model of a burglar alarm has some undesirable behaviour, for example, the sensor can switch off, or reset itself, without the control system taking note. Modify your control system Control2 definition (naming it as Control3) to remove these two problems. (4 marks)

d) It is now desired to extend the system to allow for a number of sensors to be hooked up to the control system. The process BurglarAlarm3, below, is an attempt at this. However, there are major flaws in that the control actions of the sensors are not appropriately “connected” to the control system.

$$\text{const N = 2}$$  
$$\text{range Sensors} = 0..N$$  

$$\text{\text{||SensorCollection} = ([Sensors]:Sensor).}$$  
$$\text{\text{||BurglarAlarm3} = \{ \text{Control3 || SensorCollection} \}.$$  

Modify the control system (producing Control4) to ensure that only the sensor that gets triggered can be reset, then provide some relabelling for the SensorCollection process to ensure that all sensors in the collection get switched on or off together via the on or off action in your Control4 process. (6 marks)

e) On user trials, it is realised that once a sensor has been triggered and an alarm is sounding, a user must enter the code twice in order to disable the system. The first entry of the code enables the system to reset the sensor, and the second code entry switches the whole system off.

Outline a modification to the model that will ensure only one entry of the code is necessary to disarm the whole system once a sensor has been triggered and the alarm is sounding. (4 marks)
2. 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) FSP processes may have equivalent completed trace sets but not be strongly bisimilar. Give an example of two FSP processes that exhibit that property. Hence or otherwise, give a third FSP process that can be used to distinguish the first two processes via completed trace equivalence, i.e. its composition with each will yield processes with different trace sets. Justify your answer. (4 marks)

c) Provide and explain an algorithm for computing whether two given FSP processes are strongly bisimilar. (6 marks)

d) Given the following FSP definitions

\[ P = (a \rightarrow (b \rightarrow Q) | c \rightarrow Q), \]
\[ Q = (c \rightarrow P | a \rightarrow b \rightarrow P), \]
\[ R = (a \rightarrow S | c \rightarrow R), \]
\[ S = (b \rightarrow R | b \rightarrow T), \]
\[ T = (a \rightarrow S | c \rightarrow T). \]

use the algorithm you gave in answer to part 2c above to determine whether the process \( R \) is strongly bisimilar to \( Q \). (4 marks)

e) Hence or otherwise, write down a minimal FSP process that is that is strongly bisimilar to both \( R \) and \( Q \). (2 marks)
Consider the following FSP process definitions simplistically modelling a train, gate and signal box. The open and close actions for the gate are used from the viewpoint of the train, i.e. the gate is normal closed for the train.

\[
\text{Gate} = ( \text{open} \rightarrow \text{close} \rightarrow \text{Gate} ).
\]
\[
\text{Signalbox} = ( \text{approach} \rightarrow \text{open} \rightarrow \text{passed} \rightarrow \text{close} \rightarrow \text{Signalbox} ).
\]
\[
\text{Train} = ( \text{chug} \rightarrow \text{approach} \rightarrow \text{passed} \rightarrow \text{Train}
\quad \mid \text{whistle} \rightarrow \text{STOP} ).
\]
\[
||\text{Railway} = ( \text{Gate} \mid\mid \text{Signalbox} )\backslash\{\text{open}, \text{close}\}.
\]
\[
||\text{System} = ( \text{Train} \mid\mid \text{Railway} ).
\]

a) Using transition rules for FSP, provide a detailed derivation of the chug transition that the above Train 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, and the two transition rules for hiding. Hence construct a detailed derivation of the approach transition that the composite process Railway can make.

(6 marks)

c) Now formally show using the transition rules that the process

\[
( (\text{passed} \rightarrow \text{Train}) \mid\mid \\
(\text{Gate} \mid\mid (\text{open} \rightarrow \text{passed} \rightarrow \text{close} \rightarrow \text{Signalbox}))\backslash\{\text{open}, \text{close}\} )
\]

can make a silent (\(\tau\)) transition to the process

\[
( (\text{passed} \rightarrow \text{Train}) \mid\mid \\
(\text{close} \rightarrow \text{Gate}) \mid\mid (\text{passed} \rightarrow \text{close} \rightarrow \text{Signalbox})\backslash\{\text{open}, \text{close}\} )
\]

and then make a passed transition to the process

\[
( \text{Train} \mid\mid ((\text{close} \rightarrow \text{Gate}) \mid\mid (\text{close} \rightarrow \text{Signalbox})\backslash\{\text{open}, \text{close}\} )
\]

(8 marks)

d) Hence or otherwise, build and draw a minimised labelled transition system that corresponds to the composite FSP process System.

(2 marks)
4. On concurrency concepts and concurrency in Java.

a) Briefly and clearly explain the following concepts:

i) Semaphores
ii) Monitors
iii) Threads in Java
iv) Synchronous and asynchronous message passing

(8 marks)

b) i) By reference to a simple example of a counter, modelled as a shared variable with operations to increment and decrement, which might be shared amongst several users, illustrate and explain how a monitor can be modelled in FSP.

(6 marks)

ii) Now provide an implementation of your shared counter in Java, carefully explaining its correspondence to your FSP model.

(6 marks)

5. 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

\[
\text{Sender} = (\text{in } \rightarrow \text{snd } \rightarrow \text{ack } \rightarrow \text{Sender}).
\]

\[
\text{Receiver} = (\text{snd } \rightarrow \text{ack } \rightarrow \text{out } \rightarrow \text{Receiver}).
\]

\[
\text{Composition} = (\text{Sender} \parallel \text{Receiver})\{\text{snd, ack}\}.
\]

Write down and justify what FSP process, let it be named \text{Check}, should be constructed in order to check whether the property \( P \), given above in Q5b, holds for the
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)