Concurrency and concurrent programming

Problem Sheet 1

18th October 1989
(Hand in Questions 4,5 & 6 on 25th October)

1. Design a coffee dispenser and describe its operation using CSP. The dispenser should have the following attributes:

(i) In its initial state, it requires one penny to produce a cup of coffee;
(ii) Each time a cup is produced, coffee becomes one penny more expensive;
(iii) After costing three pence, the price returns to one penny;
(iv) When an appropriate number of pennies have been inserted, a button must be pressed to produce coffee;
(v) If the wrong number of pennies are inserted, and the button is pressed, the coins are rejected;
(vi) In any event, if more than three pennies are inserted, the coins are rejected.

2. Design a drinker process who knows only that coffee might cost anything from 1-3p.

3. Show the interactive behaviour of the drinker and the coffee dispenser when the dispenser in the states ready to receive: 1p, 2p and 3p.

4. Use CSP to design a lift process. The lift moves between three floors, can be hailed by means of a button on the outside of the lift door. Once inside, a user can push a button to direct the lift to a particular floor after the doors have closed. Make notes of any extra design decisions you make in specifying the process.

5. Use CSP to describe a lift traveller who:
1. enters the building at floor 1;
2. makes for floor 3 by the lift;
3. leaves the lift, enters his office and sits at his desk.

6. Show (with explanation) the interaction between lift and office worker, when the lift is initially at floor 2.

Concurrency and concurrent programming

Problem Sheet 2

25th October 1990

(Hand in Questions 1, 2,4 & 5 on 1st November)

1. Design a transmitter and receiver process that takes inputs from two input channels, in1 and in2, and then outputs them to a single channel to a receiver so that the receiver may ensure that inputs from in1 will be output to the corresponding output channel out1, and that inputs from in2 will be output to the corresponding output channel out2. The inputs may be received arbitrarily from either channel.

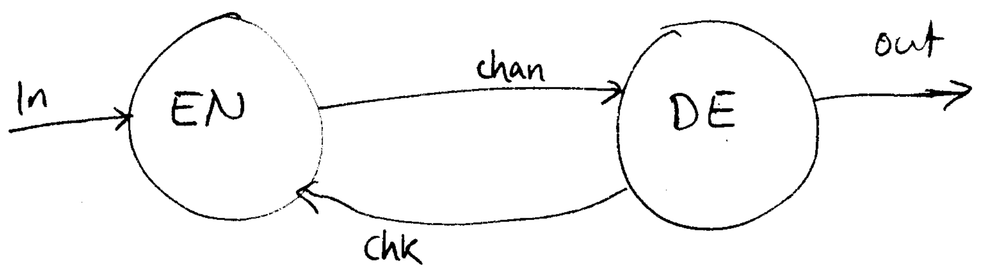
2. Prove using properties of || that the receiver and transmitter successfully achieve the correct directing of messages between channels.

3*. A single channel may cause a bottleneck if either output channel is connected to a consumer which is not consuming messages for any reason. Show how an additional channel from the receiver to the transmitter can be used to control the flow of messages, and thereby alleviate the bottleneck problem.

4. A wire between two processes may drop 1 bits, i.e. change them to 0 bits. Write an encoder and a decoder process that guarantees that a stream of bits will pass through the system without loss of information. Hint: you will require in addition a channel from the decoder to the encoder that can be used to send check information.

5 Show by means of a case argument that your protocol of question 4 works (an argument will be required that deals with the cases of correct and incorrect passage separately).

* you may find the question marked by * difficult.



```
EN =
  in → chan → EN | chk → IF x≠y THEN
                              chan → EN

DE = chan? →
  IF  x = 0 THEN  chk!x → chan?y → out!y → DE
                ELSE  out!y → DE
```

Concurrency and concurrent programming

Problem Sheet 3

6th November 1990
(Hand in all Questions by 13th November)

1. Describe Dekker's algorithm. Compare and contrast it with a semaphore solution.

2. Two user processes U1 and U2 both make use of two resource processes (P a printer and V a display unit). The printer supports channels, or events, to acquire and release the printer as well as to print messages. The display unit supports channels or events to acquire and release the display unit as well as to display messages, and move the cursor to specified co-ordinate positions. Specify the processes P and V and show that, using the rules for ||| and ||, U1 and U2 may cause deadlock, depending on how the resources are acquired and released.

What methods may be employed to avoid deadlock?

3. Redo the following question using a nondeterministic OR to define the WIRE process.

A wire between two processes may drop 1 bits, i.e. change them to 0 bits. Write a wire process and encoder and decoder process that guarantees that a stream of bits will pass through the wire correctly.

Show that your protocol is correct.

Concurrency and concurrent programming

Problem Sheet 4

13th November 1990
(Hand in all Questions by 20th November)

1. The following pair of processes computes the binomial coefficient:

```
         P                                 Q
start:   if y1 = (n-k) goto end;   start:  if y2 = k goto end
         y3 := y3 * y1;                    y2 := y2 + 1;
         y1 := y1 - 1;                     loop until y1 + y2 <= n;
         goto start;                       y3 := y3 / y2;
end:     halt                              goto start;
                                   end:    halt
```

Use semaphores to prevent interference.

2. (Conway's Problem) A sequence of cards, each with 80 characters, is to be read from a card reader and printed as a sequence of lines, each line with 125 characters, on a line printer. As many characters as possible should be packed into a line. The end of input is denoted by a special character, *eof* , in a card column. The following constraints apply to printing:

1. An extra *blank* is to be inserted at the end of every card.
2. A pair of consecutive asterisks (**) appearing in a card is to be replaced by a single *uparrow*. (A precise formulation of this requirement is as follows: a run of 2xk consecutive asterisks in a card are replaced by k *uparrows* and a run of 2xk+1 consecutive asterisks in a card are replaced by k *uparrows* followed by a single asterisk.)
3. The *eof* character is not to be printed.
4. The last output line is padded with extra blanks, if necessary to fill the line.

Use CSP to produce a collection of interacting processes to solve the problem. The solution should be made as parallel as possible.

3. Define traces (SEM) where SEM = acquire -> release -> SEM.

4. Define traces(GR) where GR = (heathrow -> GR | victoria -> GR | in10p -> euston -> GR).

Concurrency and concurrent programming

Problem Sheet 5

22nd November 1990
(Hand in all Questions by 3rd December)

1. Define traces(COPY) where COPY = in -> out -> COPY | off -> STOP.

2. Define traces (R) where R = a -> R1 where R1 = g -> d - R1 | r -> R.

3. Define traces(GR) where GR = (heathrow -> GR | victoria -> GR | in10p -> euston -> GR).

4. Show that traces(GR||TM) = traces(μX. in10p -> euston -> X)
   where TM = in10p -> (euston -> TM) |
                   in20p -> (victoria -> TM) |
                   in50p -> (heathrow -> (out10p -> TM)).

5. Show that coin -> μX.(coin -> choc -> X | choc -> coin -> X)
   sat $0 \leq ((tr\downarrow\{coin\} - (tr\downarrow\{choc\})) \leq 2$.

Concurrency and concurrent programming

Problem Sheet 6

6th December 1990
(Hand in 1 and 3 by end of term)

1*. Given the design (Sheet 2 Q 1) of a transmitter (T) and receiver (R) process that takes inputs from two input channels, in1 and in2, and then outputs them to a single channel to a receiver so that the receiver may ensure that inputs from in1 will be output to the corresponding output channel out1, and that inputs from in2 will be output to the corresponding output channel out2. Define traces(T) and traces(R) and traces ((T||R)\αmidT).

2. (i) Develop specifications for T and R of question 1 that allow you to prove that T||R correctly channels information.
(ii) What would be the appropriate specification that T||R should specify?
(iii) Give the proof that T||R satisfies the specification.
(iv) Show informally how you would prove that blockage on out1 or out2 will cause the system to wait until the blockage is free.

3*. Show that DOUBLE sat right ≤ double*(left)

where DOUBLE = left?x -> right!(x+x) -> DOUBLE
and where double* multiplies all messages in the trace by 2.

4. Define a process FIB that has the property that FIB sat (right≤ <1,1> or (<1,1> ≤ right and right'' ≤ right' + right)).

Concurrency and concurrent programming

Problem Sheet 7

11th December 1990

1. Define appropriate AND, NOT and OR processes, and by using event renaming,4. show that de Morgan's law holds for these processes, i.e. not (x and y) = not x or not y.

2. Show that COPY >> COPY is a buffer either using the rules given for pipes or traces.

3. If P and Q are buffers prove that (left?x -> (P >> (right!x -> Q))) is a buffer.

4. Do the complete transformation for the example 1 on page 13 of the printed notes.

5. Prove the special expansion rules for pipes using the definition of pipe interaction in terms of event renaming and hiding.

6. Use recursive subordinate processes to define a process that produces the Fibonacci value of a number, i.e. the process produces:

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n) &= fib(n-1) + fib(n-2) && \text{for } n>1 \end{aligned}$$