Communicating Sequential Processes
Exercises 2: Traces and Parallelism
Leila M. A. Silva

1. Give the traces of the processes below. Use the ProBE tool to confirm your answer.
   1. *a-> STOP | b-> (c->STOP | d->STOP)*
   2. *a->b->SKIP [] a-> c-> SKIP*
   3. *a->b->SKIP || a-> c-> SKIP*

2. What are the traces of the process *Copy = in?x → out !x →Copy*? What is the (strongest) traces specification of *Copy*? Use the proof rules for sat to show *Copy* satisfies that traces specification

3. A machine, *Change*, with alphabet repeatedly gives change for 5p. Its customers may choose any combination or sequence of 2p and 1p coins, provided the total value equals 5p. Construct the process *Change*.

Machine *Ch* is similar to machine *Change*, except that it will hold at most 30 1p coins and at most 17 2p coins; either of its coinstacks may be refilled at any time. Construct process *Ch* as the concurrent combination of some processes.

4. Consider the processes *P* and *Q*, defined as follows:

α P = {a, x , c, z}
α Q = {b, y, c, z}
P = (a → c → P | x → z → P)
Q = (b → c → Q | y → z → Q).

a. Rewrite *P[α P || α Q]Q* without using parallelism. Each step must be justified by one of the laws of CSP.
b. Explain four cases in which deadlock can arise in *P[α P || α Q]Q*.
c. Constuct a process *S* with alphabet *{a, x , b, y}* so that *(P[α P || α Q]Q) [α P ⋃ α Q || α S] S* behaves as much like *P[α P || α Q]Q* as possible, but does not deadlock.
d. Prove absence of deadlock by rewriting the parallel process of item c, without paralelism.
e. Use FDR to verify the presence of deadlock in *the* parallel process of item a and the absence in the parallel process of item c.

5. Let *V = (coin → choc → V) |~| (coin → toffee → V)* with the obvious alphabet.
Define the following processes:
a. *TCust* , a customer who always chooses toffee.
b. *WCust* , a customer who chooses choc or toffee at whim, possibly a different choice on each occasion.
c. *ACust* , a customer who is willing to accept whichever the machine gives.

Translate the following to a form using →, |, |~| and general recursion, but not parallelism, using the laws of CSP :

A. *V[A||T]TCust*
B. *V[A||W]WCust*
C. *V[A||AC]ACust* .

Explain why deadlock can or cannot occur in each case. Express the processes using generalized parallel composition and the FDR tool to check deadlock.

6. Two children share a paint box and an easel. Whenever they wish to paint, they first search for the easel and the box, in any order, until both are found. After they have finished painting, they drop the box and the easel. The easel and the box can each be held only by one child at a time. Write a CSP process to express the painting activity of both children. Can this system deadlock? Use FDR to check your specification. If your specification deadlocks, improves it to avoid this problem.