Exercises

Concurrency and Distributed Systems

January 2023

Contents

Vending machines

Consider the following processes, each of which describes a different kind of vending machine user:

```
Person1 =
  coin ->
    (coffee -> Person1 [] tea -> Person1)

Person2 =
  coin ->
    (coffee -> Person2 |~| tea -> Person2)

Person3 =
  coin -> coffee -> Person3
```

What happens if we place each of these in parallel with each of the vending machines `VM1`, `VM2`, and `VM3`?

In the third case, what difference does it make if we include, or do not include, `tea` in the alphabet of the user process?

Abstract

```
P =
  let
    P1 = a -> P2 [] b -> P3
    P2 = e -> P1
    P3 = f -> P1
  within
    P1

Q =
  let
    Q1 = c -> Q2 [] d -> Q3
    Q2 = e -> Q1
    Q3 = f -> Q1
  within
    Q1
```

Suppose that

```
alphaP = {a,b,e,f}
alphaQ = {c,d,e,f}

PQ = P [ alphaP || alphaQ ] Q
```

Is the process PQ deadlock free?

```
alphaPQ = {a,b,c,d,e,f}
alphaR  = {a,b,c,d}

PQR = PQ [ alphaPQ || alphaR ] R
```

Write a definition for R that will ensure that PQR is deadlock free. Explain how it achieves this.

Define a process `S` so that the assertion

```
assert S [FD= PQR
```

checks that the whole set of events `{a,b,c,d}` is available initially—that is, before any events have happened—and also available immediately after any occurrence of event `e` or event `f`.