6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	KavinduWanasekara
Student ID	20191067
Date	12/01/2023

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING_SYSTEM
Description	Models a printer machine, a technician and two students
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	Process: PRINTING_SYSTEM Alphabet: { s2.acquirePrint,s2.acquireRefill, s2.print, s2.refill, s2.release,s2.wait,s3.acquirePrint, s3.acquireRefill, s3.print, s3.refill, s3.release, s3.wait, t.acquirePrint, t.acquireRefill, t.print, t.refill, t.release, t.wait }
Sub-processes (List them.)	STUDENT, TECHNICIAN, PRINTER
Number of States	55
Deadlocks (yes/no)	No
Deadlock Trace(s) **(If applicable)**	Not Required

2. FSP "main" Program Code

The code for the parallel composition of all of the sub-processes and the definitions of any constants, ranges & process labelling sets used. (Do not include the code for the other sub-processes.)

FSP Program:

```
const MAX_SHEETS = 3 // Maximum number of sheets in a printer

set PRINT_ACTIONS = {acquirePrint, print, acquireRefill, refill, release}

// Initialise printer with given number of sheets and continue
PRINTER (PAPER_COUNT = MAX_SHEETS) = PRINTER[PAPER_COUNT],
PRINTER[p : 0..PAPER_COUNT] =
        if (p > 0)
        then (acquirePrint -> print -> release -> PRINTER[p-1])
        else (acquireRefill -> refill -> release -> PRINTER[MAX_SHEETS]).

// Initialise student and print given number of documents
STUDENT(DOCUMENT_COUNT = 1) = STUDENT[DOCUMENT_COUNT],
STUDENT[document : 1..DOCUMENT_COUNT] = (
        acquirePrint -> print[document] ->
                if (document > 1)
                then (release -> STUDENT[document-1])
                else (release -> END) | // Prints the last document
                wait -> STUDENT
) + PRINT_ACTIONS / {print/print[document:1..DOCUMENT_COUNT]}.

TECHNICIAN = (
        acquireRefill -> refill -> release -> TECHNICIAN |
        wait -> TECHNICIAN
) + PRINT_ACTIONS.

||PRINTING_SYSTEM = (s3: STUDENT(3) || s2: STUDENT(2) || t: TECHNICIAN || {s3, s2, t} ::
PRINTER).
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
STUDENT	The process describes students who print documents. A student can print three documents at once

TECHNICIAN	The process describes a technician refill the printer when paper count become zero
PRINTER	The process act as a resource to student and technician. Printer performs print process and refill process in each requirement.

4. Analysis of Combined Process Actions

- **Synchronous** actions are performed by at least two sub-process in the combination.
- **Blocked Synchronous** actions cannot be performed, since at least one of the sub-processes cannot preform them, because they were added to their alphabet using alphabet extension.
- **Asynchronous** actions are preformed independently by a single sub-process.

Group actions together if appropriate, for example if they include indexes,
e.g. in[0], in[1], …, in[5] as in[1..5].

(Add rows as necessary.)

Synchronous Actions	Synchronised by Sub-Processes (List)
s2.acquirePrint, s2.print, s3.acquirePrint, s3.print,	STUDENT, PRINTER
t.acquireRefill, t.refill,	TECHNICIAN, PRINTER
s2.release, s3.release, t.release	STUDENT, PRINTER, TECHNICIAN

Sub-Process	Asynchronous Actions (List)
TECHNICIAN	t.wait
PRINTER	None
STUDENT	S2.wait, s3.wait

5. Parallel Composition Structure Diagram