6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Wehan Withana
Student ID	2019349
Date	2023/01/12

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING_SYSTEM
Description	It’s a model between 2 students, 1 technician and one printer. One student prints two documents and other student prints 3 documents. Technician refill the paper when printer out of papers.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	{ stu1.acquire, stu1.empty, stu1.print[1], stu1.print[2], stu1.print[3], stu1.refill_printer, stu1.release, stu2.acquire, stu2.empty, stu2.print[1], stu2.print[2], stu2.print[3], stu2.refill_printer, stu2.release, techn.empty, techn.print[1], techn.print[2], techn.print[3], techn.refill_printer, techn.release, terminate }
Sub-processes (List them.)	PRINTER, STUDENT, TECHNICIAN
Number of States	56
Deadlocks (yes/no)	No Deadlocks
Deadlock Trace(s) (If applicable)	None

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 MIN_SHEET             =     1
const MAX_SHEET             =     3
range DOCUMENT_COUNT        =     MIN_SHEET .. MAX_SHEET
range SHEET_STACK           =     0 .. MAX_SHEET


set USER_SET = {stu1, stu2, techn}
set ACTIONS_SET = {acquire, print[DOCUMENT_COUNT], release, empty}


|| PRINTING_SYSTEM = (stu1: STUDENT(2) || stu2: STUDENT(3) || techn : TECHNICIAN ||
USER_SET :: PRINTER)
/ {terminate/USER_SET.terminate}.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
PRINTER	Its represent a basic printer which has three print sheets at once
STUDENT(2)	Depicts a student attempting to have two documents printed
STUDENT(3)	Depicts a student attempting to have three documents printed
TECHNICIAN	A technician who refills 3 papers when printer out of papers

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)
stu1.acquire, stu1.print[1], stu1.print[2], stu1.release	STUDENT(2), PRINTER
stu2.acquire, stu2.print[1], stu2.print[2], stu2.print[3], stu2.release	STUDENT(3), PRINTER
techn.empty, techn.refill_printer, techn.release	TECHNICIAN, PRINTER
terminate	STUDENT(2), STUDENT(3), TECHNICIAN

Sub-Process	Asynchronous Actions (List)
TECHNICIAN	techn.refill_printer
PRINTER	None
STUDENT(2)	None
STUDENT(3)	None

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.