6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Ihan Dilnath Lelwala
Student ID	2016030 | W1673607
Date	January 2021

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING_SYSTEM
Description	This composite process models a Printing System. It consists of 4 subprocesses: 2 Student processes, 1 Technician process and 1 Printer process which is shared among the Student and Technician processes. It facilitates mutual exclusive access to an user who wants to operate the shared Printer.

Alphabet
(Use LTSA's compressed notation, if alphabet is large.)
	alphabet (PRINTING_SYSTEM) = {
	student1.{{acquireToPrint, acquireToRefill, cannotFill, fill},
	printDocument[1..3], release},
	student2.{{acquireToPrint, acquireToRefill, cannotFill, fill},
	printDocument[1..2], release},
	technician.{acquireToPrint, acquireToRefill, cannotFill, fill, release},
	terminate
	}

Sub-processes (List them.)	PRINTER, student1 : STUDENT(3), student2 : STUDENT(2) TECHNICIAN
Number of States	80
Deadlocks (yes/no)	No
Deadlock Trace(s) **(If applicable)**	Not Applicable

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:

```
range PAPER_TRAY = 0..3
const EMPTY_PAPER_TRAY = 0
const FULL_PAPER_TRAY = 3

set Students = { student1, student2 }
set Users = { Students, technician }

// Actions that should not be performed by Students and Technicians separately
set StudentProhibitedActions = { acquireToRefill, fill, cannotFill }
set TechnicianProhibitedActions = { acquireToPrint }

||PRINTING_SYSTEM = ( student1: STUDENT(3) || student2: STUDENT(2) ||
technician: TECHNICIAN || Users :: PRINTER ) / { terminate / Users.terminate }.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
PRINTER	The Printer process models the behaviour of the printer including the paper tray state.
student1 : STUDENT(3)	The Student process models the behaviour of the students who uses the printer to print documents. This is one instance of a Student process, where the student wishes to print 3 documents.
student2 : STUDENT(2)	Same as the above. However, this student wishes to print 2 documents only.
TECHNICIAN	The Technician process models the behaviour of the technician who refills the printer with paper when it runs out of paper.

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 perform 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)
student1.{acquireToPrint, acquireToRefill, cannotFill, fill, release}	student1: STUDENT(3), PRINTER
student2.{acquireToPrint, acquireToRefill, cannotFill, fill, release}	student2: STUDENT(2), PRINTER
technician.{acquireToPrint, acquireToRefill, cannotFill, fill, release}	TECHNICIAN, PRINTER
terminate	student1: STUDENT(3), student2: STUDENT(2), TECHNICIAN

Blocked Synchronous Actions	Synchronising Sub-Processes	Blocking Sub-Processes
technician.acquireToPrint	TECHNICIAN, PRINTER	TECHNICIAN
student1.{acquireToRefill, cannotFill, fill}	student1 : STUDENT(3), PRINTER	student1 : STUDENT(3)
student2.{acquireToRefill, cannotFill, fill}	student2 : STUDENT(2), PRINTER	student2 : STUDENT(2)

Sub-Process	Asynchronous Actions (List)
student1 : STUDENT(3)	student1.printDocument[1..3]
student2 : STUDENT(2)	student2.printDocument[1..2]
PRINTER	None
TECHNICIAN	None

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.