6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Ravija Gunawardana
Student ID	W1761290 | 2019340
Date	January 2023

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING_SYSTEM
Description	This composite process models a printing system. It has four subprocesses: two for students, one for technicians, and one shared by the student and technician processes. It makes it easier for a user who wishes to utilize the shared Printer to gain mutually exclusive access.
Alphabet (Use LTSA's compressed notation, if the alphabet is large.)	{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:

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

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

set ActionsCannotPerformByStudent = { acquireToRefill, fill, cannotFill } // Student
cannont preform thease actions
set ActionsCannotPerformByTechnician = { acquireToPrint } // Paper technician
cannont preform thease actions

TECHNICIAN = (acquireToRefill -> { fill, cannotFill } -> release -> TECHNICIAN |
terminate -> END) + ActionsCannotPerformByTechnician.
||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
student1 : STUDENT(3)	The Student procedure serves as a paradigm for how students should behave while using the printer to print documents. In this particular student process, the student wants to print two documents.
student1 : STUDENT(3)	The Student procedure serves as a paradigm for how students should behave while using the printer to print documents. In this particular student process, the student wants to print three documents.
TECHNICIAN	When the printer runs out of paper, the technician will replace it according to the behavior modeled by the procedure.
PRINTER	The Printer process simulates printer activity, including the condition of the paper tray.

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

Sub-Process	Asynchronous Actions (List)
PRINTER	No Asynchronous Actions
TECHNICIAN	No Asynchronous Actions
student2 : STUDENT(2)	student2.printDocument[1..2]
student1 : STUDENT(3)	student1.printDocument[1..3]

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.