6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Thiloshon Nagarajah
Student ID	W1608486 / 2015298
Date	6 Jan 2019

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTJOB
Description	Models the process of two students printing documents with 3 and 2 pages each with technician refilling papers as required.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	{a.{print.doc[1..3], student.{acquire, release}, technician.{acquire, refill, release}}, b.{print.doc[1..2], student.{acquire, release}, technician.{acquire, refill, release}}, t.{student.{acquire, print.paper, release}, technician.{acquire, refill, release}}, waiting}
Sub-processes (List them.)	STUDENT PRINTER TECHNICIAN
Number of States	67 (55 without waiting logic)
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

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 sub-processes.)

FSP Program:

```
// CONSTANTS
const MAX_PAPER = 3
range PAPER_RANGE = 0 .. MAX_PAPER

set Students = {a, b}
set PRINTER_ACTIONS = {student.acquire , student.print.paper , student.release,
technician.acquire , technician.refill , technician.release }

// PRINT COMPOSITE FSM
|| PRINTJOB = (a:STUDENT(3) ||b:STUDENT(2) || t:TECHNICIAN || {Students, t}::PRINTER(3)
)  /
       {
              waiting / { a.waiting, b.waiting, t.waiting},
              a.print.doc[1..3] / {a.student.print.paper} , b.print.doc[1..2] /
{b.student.print.paper}
       } .
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
STUDENT	Models the student wanting to print document.
TECHNICIAN	Models the technician in charge of refilling paper as required.
PRINTER	Models the printer that can print 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 perform them, because they were added to their alphabet using alphabet extension.
- **Asynchronous** actions are preformed independently by a single sub-process.

(Add rows as necessary.)

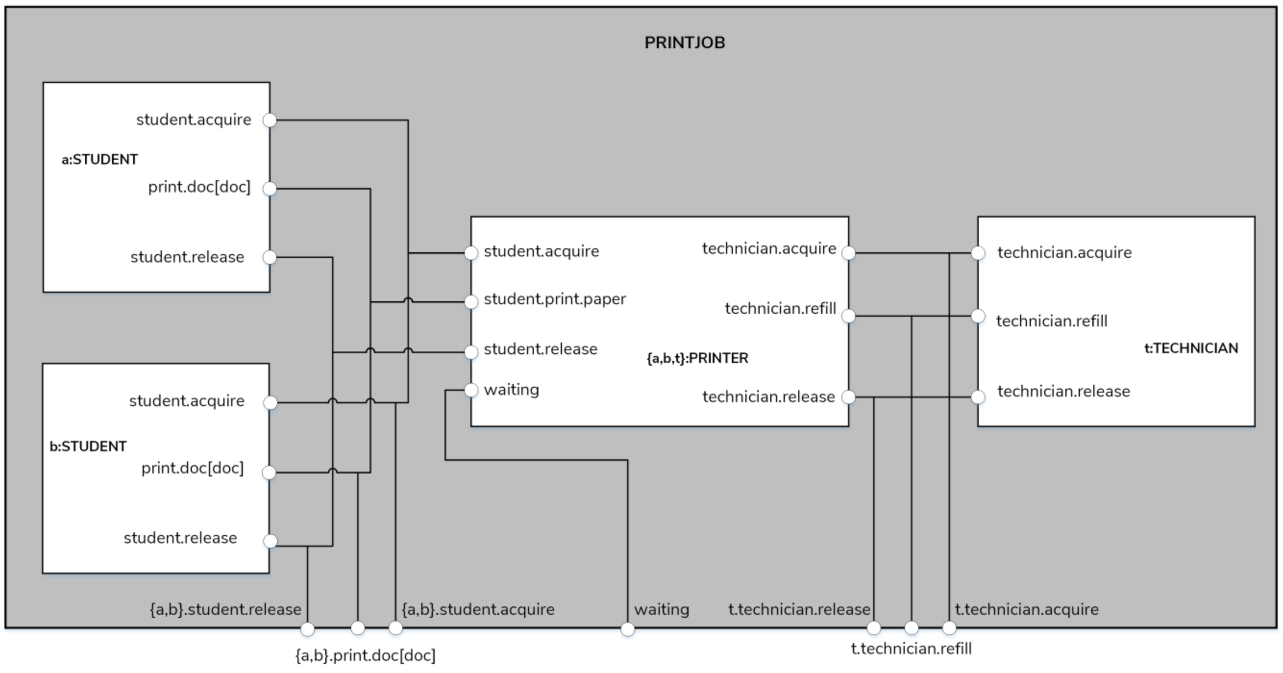
Synchronous Actions	**Synchronised by Sub-Processes (List)**
a.student.acquire, b.student.acquire, a.print.doc[1..3], b.print.doc[1..2], a.student.release, b.student.release	STUDENT , PRINTER
t.technician.acquire, t.technician.release, t.technician.refill	TECHNICIAN, PRINTER

Blocked Synchronous Actions	**Synchronising Sub-Processes (List)**	**Blocking Sub-Processes**
a.student.acquire, b.student.acquire, a.print.doc[1..3], b.print.doc[1..2], a.student.release, b.student.release	STUDENT, PRINTER	TECHNICIAN
t.technician.acquire, t.technician.release, t.technician.refill	TECHNICIAN, PRINTER	STUDENT

Sub-Process	**Asynchronous Actions (List)**
waiting	PRINTER

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


PRINTJOB
a:STUDENT
student.acquire
print.doc[doc]
student.release
b:STUDENT
student.acquire
print.doc[doc]
student.release
{a,b,t}:PRINTER
student.acquire
student.print.paper
student.release
waiting
technician.acquire
technician.refill
technician.release
t:TECHNICIAN
technician.acquire
technician.refill
technician.release
{a,b}.student.release
{a,b}.print.doc[doc]
{a,b}.student.acquire
waiting
t.technician.release
t.technician.refill
t.technician.acquire