6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTER_SYSTEM
Description	Models a printer that is shared by two students and has a technician that will refill the paper when it runs out.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	{ student_1.acquire, student_1.notEnoughPaper, student_1.outOfPaper, student_1.paperNotEmpty, student_1.print[0..3], student_1.release, student_1.servicePrinter, student_2.acquire, student_2.notEnoughPaper, student_2.outOfPaper, student_2.paperNotEmpty, student_2.print[0..3], student_2.release, student_2.servicePrinter, tech.acquire, tech.notEnoughPaper, tech.outOfPaper, tech.paperNotEmpty, tech.print[0..3], tech.release, tech.servicePrinter }
Sub-processes (List them.)	STUDENT, PRINTER, TECH
Number of States	136
Deadlocks (yes/no)	No deadlocks / errors
Deadlock Trace(s)	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 sub-processes.)

FSP Program:

```
const MAX_ID = 3
range DOC_ID = 0 .. MAX_ID
const MAX_PAPR = 3
range PAPR_LVL = 0 .. MAX_PAPR

set PRINT_Actions = { print[DOC_ID], acquire, release, outOfPaper, servicePrinter,
                      paperNotEmpty, notEnoughPaper }
set All_Users = { tech, student_1, student_2 }

|| PRINTER_SYSTEM
=      (  student_1 : STUDENT ( 3 )
       || student_2 : STUDENT ( 2 )
       || tech : TECH
       || All_Users :: PRINTER ( 3 )
       ) .
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
PRINTER	Represents printer machine that can print documents.
STUDENT	Represents a simple student that is initiated with a certain amount of documents and can then use the printer to print a document
TECH	Represents a simple technician that can refill the amount of papers the printer has once they are used by the printer

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.

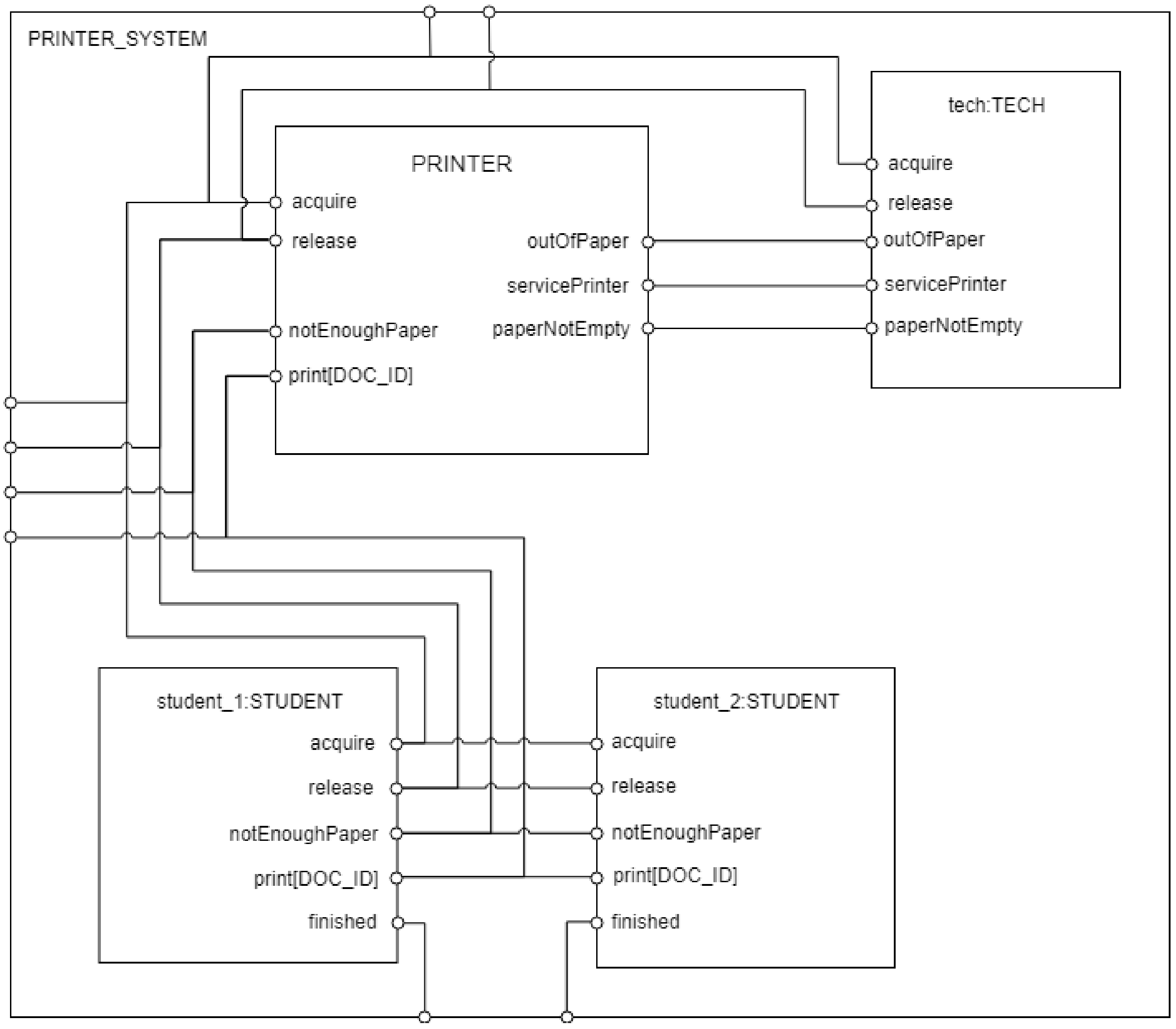
(Add rows as necessary.)

Synchronous Actions	Synchronised by Sub-Processes (List)
acquire, print[DOC_ID], release, notEnoughPaper	STUDENT, PRINTER
acquire, outOfPaper, serviceprinter, release, paperNotEmpty	TECH, PRINTER

Sub-Process	Asynchronous Actions (List)
STUDENT	finished

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


PRINTER_SYSTEM
PRINTER
acquire
release
notEnoughPaper
print[DOC_ID]
outOfPaper
servicePrinter
paperNotEmpty
tech:TECH
acquire
release
outOfPaper
servicePrinter
paperNotEmpty
student_1:STUDENT
acquire
release
notEnoughPaper
print[DOC_ID]
finished
student_2:STUDENT
acquire
release
notEnoughPaper
print[DOC_ID]
finished