6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Ashfaaq Ahamed Hilal
Student ID	2019394
Date	31/12/2022

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING_SYSTEM_COMPOSITION
Description	PRINTING_SYSTEM_COMPOSITION is a composite process. This process comprises of four primitive sub processes to form the composite process. The set “Students” has two elements as “student_one” and “student_two”. The set “Users” contains two elements as “technician” and set “Students”. The primitive processes are the STUDENT_PROCESS, TECHNICIAN_PROCESS and MY_PRINTER. STUDENT_PROCESS gets mutually exclusive control of the MY_PRINTER. TECHNICIAN_PROCESS checks paper level repeatedly of the printer and refills if its less. MY_PRINTER can contain only maximum of three papers at a time. Two STUDENT_PROCESS process are carried out by the PRINTING_SYSTEM_COMPOSITION. One prints two documents and other prints three. One TECHNICIAN_PROCESS process refills with three sheets. One MY_PRINTER process is used mutual exclusively by the STUDENT_PROCESS and TECHNICIAN_PROCESS. After the end of the printing process, all users are terminated. There is no deadlock or errors outperformed by the PRINTING_SYSTEM_COMPOSITION.
Alphabet (Use LTSA's compressed notation if alphabet is large.)	Alphabet: { student_one.{{acquireToPrint, acquireToRefill, cannotFill, fill}, printDocument[1..3], release}, student_two.{{acquireToPrint, acquireToRefill, cannotFill, fill}, printDocument[1..2], release}, technician.{acquireToPrint, acquireToRefill, cannotFill, fill, release}, terminate }

Sub-processes (List them.)	• student_one:STUDENT_PROCESS(3) • student_two:STUDENT_PROCESS(2) • TECHNICIAN_PROCESS • MY_PRINTER
Number of States	80
Deadlocks (yes/no)	No
Deadlock Trace(s) (If applicable)	No

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:

ONLY CONSTANTS, RANGE, SET, COMPOSITES. DON'T INCLUDE PROCESSES

```
range RANGE_OF_PAPER_TRAY = 0..3
const EMPTY_PAPER_TRAY_VALUE = 0
const FULL_PAPER_TRAY_VALUE = 3

// ones who acess the printer
set Students = { student_one, student_two }
set Users = { Students, technician }

// Prohibited students and technicians actions separately
set ProhibitedActionsForStudent = { acquireToRefill, fill, cannotFill }
set ProhibitedActionsForTechnician = { acquireToPrint }

||PRINTING_SYSTEM_COMPOSITION = ( student_one: STUDENT_PROCESS(3) ||
student_two: STUDENT_PROCESS(2) || technician: TECHNICIAN_PROCESS || Users
:: MY_PRINTER ) / {terminate/Users.terminate}.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
student_one:STUDENT_PROCESS(3)	A sub process of the system which uses the printer by providing documents to be printed. In this occasion, the student provides three documents to the printer to be printed. Here, the printing process takes place only if there is sufficient amount of paper in the paper tray.
student_two:STUDENT_PROCESS(2)	A sub process of the system which uses the printer by providing documents to be printed. In this occasion, the student provides two documents to the printer to be printed. Here, the printing process takes place only if there is sufficient amount of paper in the paper tray.
TECHNICIAN_PROCESS	A sub process of the system where the process checks for the paper count in the printer. Refills the printer with maximum of three sheets if the paper tray is empty. Mutually exclusively checks for the paper count and refills once the tray is empty by making the printing of document to wait.
MY_PRINTER	This sub process is used by the STUDENT and TEECHNICIAN sub processes which is used to print the documents. This also enables to check the paper level availability in the printer and allows it to be refilled.

4. Analysis of Combined Process Actions

- **Synchronous** actions are performed by at least two sub-processes 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.

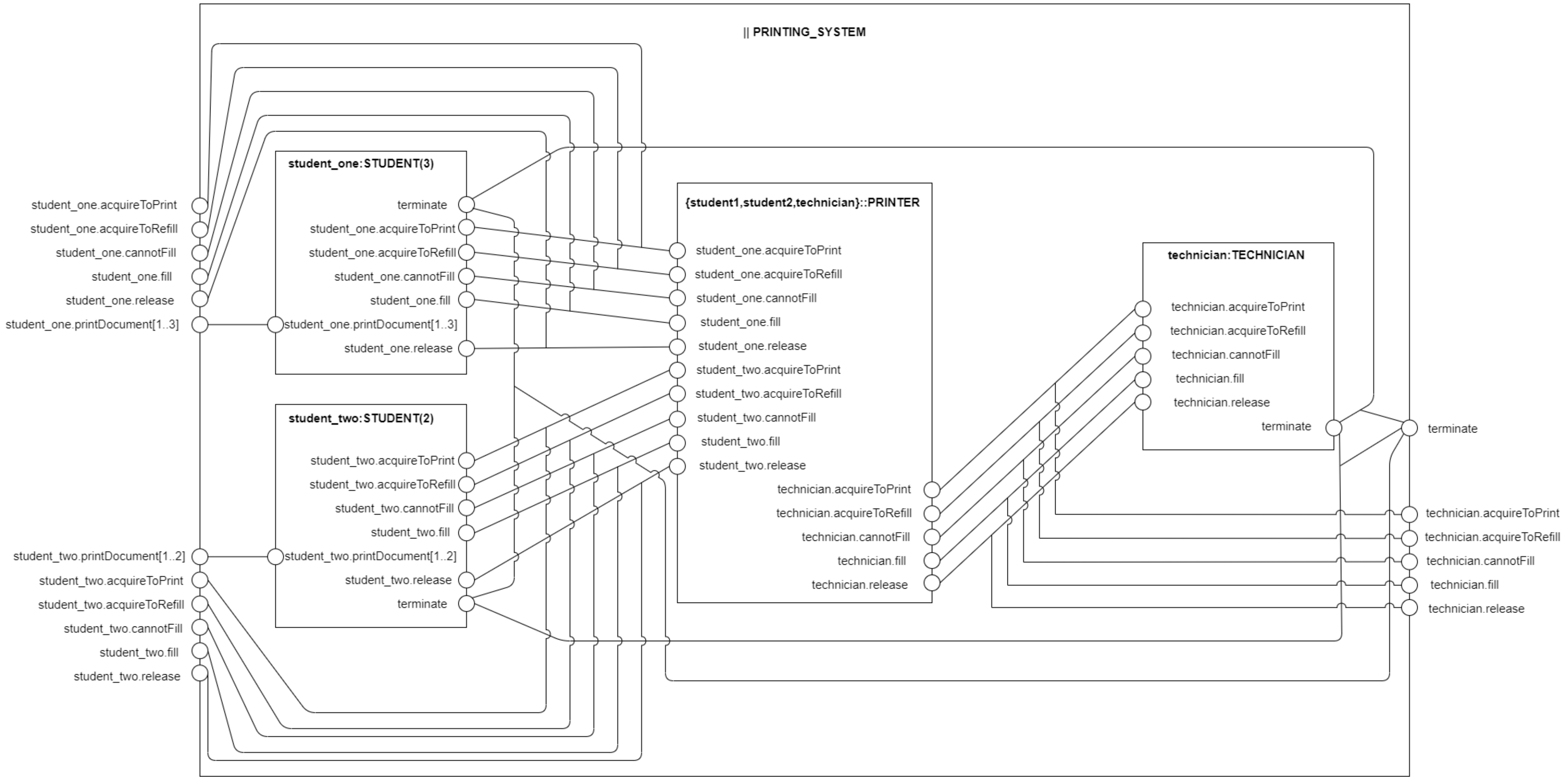
Group actions together if appropriate, for example if they include indexes,
e.g. in[0], in[1], …, in[5] as in[1..5].

Synchronous Actions	Synchronised by Sub-Processes (List)
student_one.{{acquireToPrint, acquireToRefill, cannotFill, fill}, release}	student_one:STUDENT_PROCESS(3) and MY_PRINTER
student_two.{{acquireToPrint, acquireToRefill, cannotFill, fill}, release}	student_two:STUDENT_PROCESS(2) and MY_PRINTER
technician.{acquireToPrint, acquireToRefill, cannotFill, fill, release},	TECHNICIAN_PROCESS and MY_PRINTER
terminate	student_one:STUDENT_PROCESS(3), student_two:STUDENT_PROCESS(2) TECHNICIAN_PROCESS

Sub-Process	Asynchronous Actions (List)
student_one:STUDENT_PROCESS(3)	student_one.printDocument[1..3]
student_two:STUDENT_PROCESS(2)	student_two.printDocument[1..2]
TECHNICIAN_PROCESS	No Asynchronous Actions
MY_PRINTER	No Asynchronous Actions

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


|| PRINTING_SYSTEM
student_one:STUDENT(3)
terminate
student_one.acquireToPrint
student_one.acquireToRefill
student_one.cannotFill
student_one.fill
student_one.printDocument[1..3]
student_one.release
student_two:STUDENT(2)
student_two.acquireToPrint
student_two.acquireToRefill
student_two.cannotFill
student_two.fill
student_two.printDocument[1..2]
student_two.release
terminate
{student1,student2,technician}::PRINTER
student_one.acquireToPrint
student_one.acquireToRefill
student_one.cannotFill
student_one.fill
student_one.release
student_two.acquireToPrint
student_two.acquireToRefill
student_two.cannotFill
student_two.fill
student_two.release
technician.acquireToPrint
technician.acquireToRefill
technician.cannotFill
technician.fill
technician.release
technician:TECHNICIAN
technician.acquireToPrint
technician.acquireToRefill
technician.cannotFill
technician.fill
technician.release
terminate
student_one.acquireToPrint
student_one.acquireToRefill
student_one.cannotFill
student_one.fill
student_one.release
student_one.printDocument[1..3]
student_two.printDocument[1..2]
student_two.acquireToPrint
student_two.acquireToRefill
student_two.cannotFill
student_two.fill
student_two.release
terminate
technician.acquireToPrint
technician.acquireToRefill
technician.cannotFill
technician.fill
technician.release