6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Asel Siriwardena
Student ID	2017454
Date	12/27/2020

1. FSP Composition Process Attributes

Attribute	**Value**
Name	PRINTER_SYSTEM
Description	The composite process of the printer, the two students and the technician. One of the students (s1) is trying to print two documents, while the other student (s2) is trying to print three documents. Technician (tcn) refills the printer when the printer runs out of paper.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	{{acquire, empty}, print[1..3], {refill_printer, release}, {s1, s2}.{{acquire, empty}, print[0..3], {refill_printer, release}}, start, tcn.{{acquire, empty}, print[1..3], {refill_printer, release, wait}}}
Sub-processes (List them.)	STUDENT(1), STUDENT(2), TECHNICIAN, PRINTER
Number of States	56
Deadlocks (yes/no)	No deadlocks/errors
Deadlock Trace(s) **(If applicable)**	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 other sub-processes.)

FSP Program:

```
/*  Author: Asel Siriwardena
        2017454 | w1698419
*/


// All the users of the system
set All_USERS = {s1, s2, tcn}


// Composite printer process [ with students, technician ]
|| SHARED_PRINTER = ( s1: STUDENT(2) || s2: STUDENT(3) || tcn : TECHNICIAN ||
PRINTER )
/ {start/s1.start,start/s2.start}.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
PRINTER	Represents a simple printer which can hold three sheets of a time
STUDENT(2)	Represents a student who is trying get two documents printed
STUDENT(3)	Represents a student who is trying get three documents printed
TECHNICIAN	Represents a technician who refills the printer when the printer runs out of paper (i.e. refills three papers at a time)

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.)

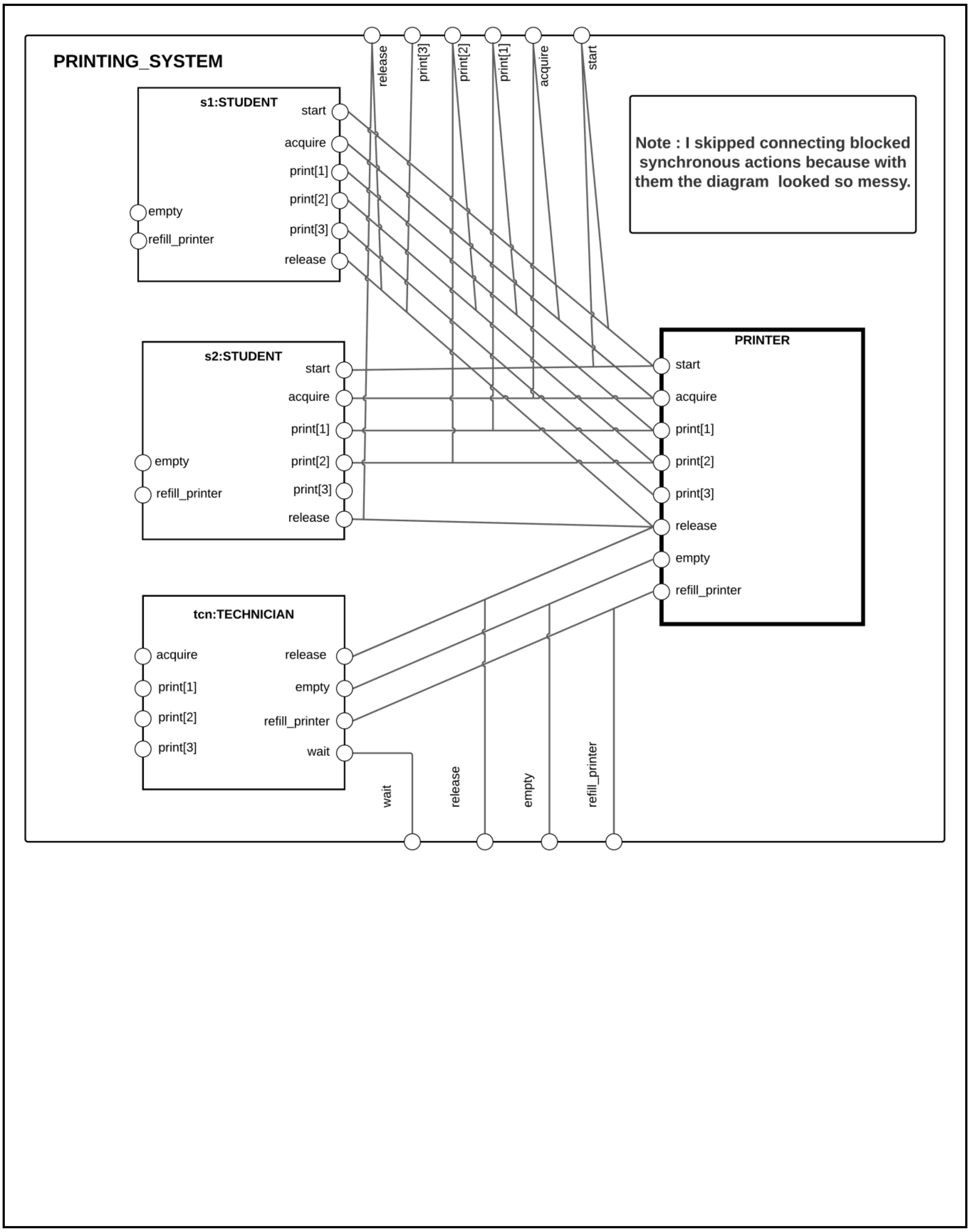
Synchronous Actions	**Synchronised by Sub-Processes (List)**
start	STUDENT(2), STUDENT(3), PRINTER
s1.acquire, s1.print[1], s1.print[2], s1.release	STUDENT(2), PRINTER
s2.acquire, s2.print[1], s2.print[2], s2.print[3], s2.release	STUDENT(3), PRINTER
tcn.empty, tcn.refill_printer, tcn.release	TECHNICIAN, PRINTER

Sub-Process	**Asynchronous Actions (List)**
TECHNICIAN	tcn.wait
PRINTER	None
STUDENT(2)	None
STUDENT(3)	None

Blocked Synchronising Actions	**Synchronising Sub-Processes**	**Blocking Sub-Processes**
tcn.print[1], tcn.print[2], tcn.print[3], tcn.acquire	TECHNICIAN, PRINTER	TECHNICIAN
s1.empty s2.refill_printer	STUDENT(2), PRINTER	STUDENT(2)
s2.print[3] s2.empty s2.refill_printer	STUDENT(3), PRINTER	STUDENT(3)

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


PRINTING_SYSTEM
release
print[3]
print[2]
print[1]
acquire
start
s1:STUDENT
start
acquire
print[1]
print[2]
print[3]
release
empty
refill_printer
Note : I skipped connecting blocked synchronous actions because with them the diagram looked so messy.
PRINTER
start
acquire
print[1]
print[2]
print[3]
release
empty
refill_printer
s2:STUDENT
start
acquire
print[1]
print[2]
print[3]
release
empty
refill_printer
tcn:TECHNICIAN
acquire
print[1]
print[2]
print[3]
release
empty
refill_printer
wait
wait
release
empty
refill_printer