6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	J. M. D Dilan Dicman
Student ID	2019284
Date	12/01/2023

1. FSP Composition Process Attributes

Attribute	**Value**
Name	SHARED_PRINTER
Description	This model defines the shared system of printing system. Some of the actions are shared among the processes. SHARED_PRINTER composite process includes 2 STUDENT processes, 1 TECHNICIAN process, and 1 PRINTER process. PRINTER process is the process that is shared among the other mentioned processes.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	alphabet(SHARED_PRINTER) = { s1.{{aquireLock, out_of_paper, print}, print[0..2], {refill, releaseLock, terminate}}, s2.{{aquireLock, out_of_paper, print}, print[0..1], {refill, releaseLock, terminate}}, t1.{aquireLock, out_of_paper, print, refill, releaseLock, skip, terminate}}
Sub-processes (List them.)	STUDENT(3), STUDENT(2), STUDENT(1)
Number of States	20
Deadlocks (yes/no)	yes
Deadlock Trace(s) **(If applicable)**	Trace to DEADLOCK: s1.aquireLock s1.print.0 t1.terminate

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 MAX_SHEETS = 3 const MAX_DOCS = 3 range PrintRange = 0..MAX_SHEETS range StuPrintRange = 0..MAX_DOCS set PrintActions = {aquireLock, refill, print, releaseLock, out_of_paper, terminate} || SHARED_PRINTER = (s1:STUDENT(3) || s2:STUDENT(2) || t1:TECHNICIAN || {s1, s2, t1} :: PRINTER).

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
PRINTER	Models the printer process which defines the print action and refill action.
s1: STUDENT(3)	Models the Student process which is defined to print documents according to the number of docs passed as the parameter
s2: STUDENT(2)	Same STUDENT process which print 2 documents
TECHNICIAN	The technician checks for empty tray and refills the sheets

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].

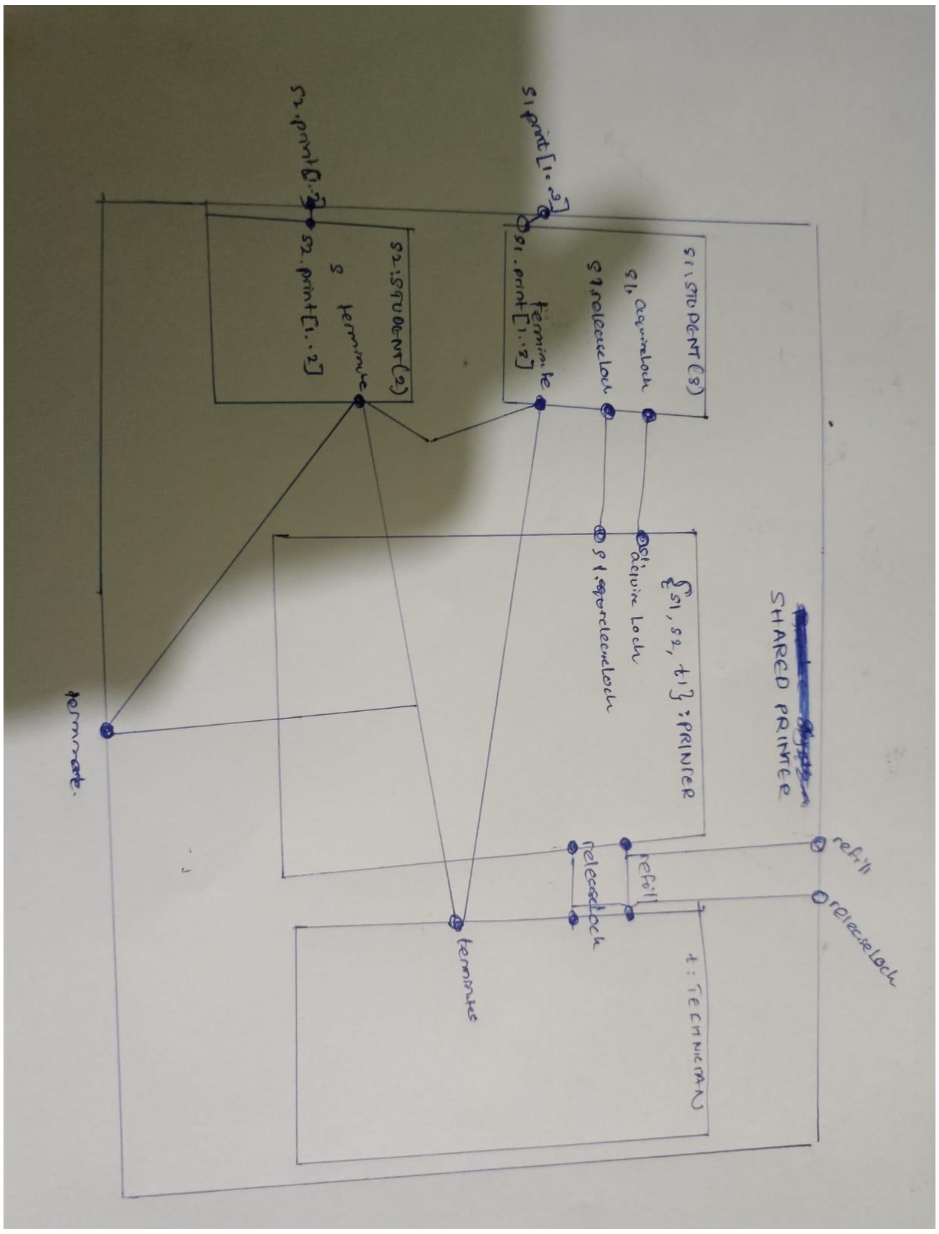
(Add rows as necessary.)

Synchronous Actions	**Synchronised by Sub-Processes (List)**
S1.{acquireLock, releaseLock}	S1: STUDENT(3), PRINTER
S2.{acquireLock, releaseLock}	S2: STUDENT(2), PRINTER
T1. {refill, releaseLock}	TECHNICIAN, PRINTER
terminate	S1:STUDENT(3), s2: STUDENT(2), TECHNICIAN

Sub-Process	**Asynchronous Actions (List)**
S1:STUDENT(3)	S1.print[1..3]
S2.STUDENT(2)	S2.print[1..2]
PRINTER	None
TECHNICIAN	None

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


s1.print[1..3]
s2.print[1..2]
s1:STUDENT(3)
s1.acquirelock
s1.releaselock
s1.print[1..3]
terminate
s2:STUDENT(2)
s
terminate
s2.print[1..2]
terminate
SHARED PRINTER
{s1, s2, t1}::PRINTER
s1.acquire lock
s1.releaselock
refill
releaselock
refill
releaseLock
t:TECHNICIAN
terminate