6SENG006W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Harini Rodrigo
Student ID	2019754
Date	19/12/2023

1. FSP Composition Process Attributes

Attribute	**Value**
Name	TICKET_MACHINE
Description	Composition process containing 5 subprocesses. Three Passenger (pas1, pas2, pas3) and two technicians (paperTech, tonerTech) are the five subprocesses.

Sub-processes (List them.)	pas1.PASSENGER(1)
	pas2.PASSENGER(1)
	pas3.PASSENGER(1)
	paperTech.PAPER_TECH
	tonerTech.TONER_TECH
	TICKET_PRINTER

Number of States	44
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 individual sub-processes.)

FSP Program:

```
// Define the set of technicians (TEC) and passengers (PAS)
set TEC = {paperTech, tonerTech}
set PAS = {pas1, pas2, pas3}

// Define constant values for machine parameters
const MAX_PAPER_LEVEL = 3
const MIN_PAPER_LEVEL = 0
const MIN_TONER_LEVEL = 0
const MAX_TONER_LEVEL = 3
const TICKET_COUNT = 3

// Define ranges for paper and toner levels
range PAPER_RANGE = MIN_PAPER_LEVEL..MAX_PAPER_LEVEL
range TONER_RANGE = MIN_TONER_LEVEL..MAX_TONER_LEVEL

// Parallel composition of processes to form the TICKET_MACHINE system
|| TICKET_MACHINE = (
    pas1: PASSENGER(1) / {exit / pas1.exit}
    || pas2: PASSENGER(1) / {exit / pas2.exit}
    || pas3: PASSENGER(1) / {exit / pas3.exit}
    || paperTech: PAPER_TECH / {exit / paperTech.exit}
    || tonerTech: TONER_TECH / {exit / tonerTech.exit}
    || {PAS,TEC}::TICKET_PRINTER / //Define transition
    {
        // Action relabeling
        paperTech.acquirePaperRefill / {tonerTech.acquirePaperRefill,
PAS.acquirePaperRefill},
        paperTech.paperRefill / {tonerTech.paperRefill, PAS.paperRefill},
        paperTech.releasePaperRefill / {tonerTech.releasePaperRefill,
PAS.releasePaperRefill},
        tonerTech.acquireTonerRefill / {paperTech.acquireTonerRefill,
PAS.acquireTonerRefill},
        tonerTech.tonerRefill / {paperTech.tonerRefill, PAS.tonerRefill},
        tonerTech.releaseTonerRefill / {paperTech.releaseTonerRefill,
PAS.releaseTonerRefill},
        PAS.acquireTicketMachine / TEC.acquireTicketMachine,
        PAS.printTicket / TEC.printTicket,
        PAS.releaseTicketMachine / TEC.releaseTicketMachine
    }
).
```

3. Combined Sub-processes
(Add rows as necessary.)

Process	Description
pas1.PASSENGER(1)	The pas1 process simulates the actions of a passenger, representing the behaviour of an individual who requests access to the shared ticket machine to print a ticket.
pas2.PASSENGER(1)	The pas2 process emulates the behaviour of another passenger, similar to pas1, by requesting access to the shared ticket machine for the purpose of printing a ticket.
pas3.PASSENGER(1)	The pas3 process mirrors the behaviour of another passenger. It represents the actions of a passenger who seeks access to the shared ticket machine to print a ticket.
paperTech.PAPER_TECH	The paperTech process captures the activities of the paper technician. This role involves the behaviour of a technician responsible for the paper refilling operation.
tonerTech.TONER_TECH	The tonerTech process delineates the behaviour of the toner technician. It represents the actions of a technician responsible for the toner refilling operation.
TICKET_PRINTER	The TICKET_PRINTER process encapsulates the behaviour of the ticket printer. This process coordinates the actions related to ticket printing, paper management, and toner levels within the shared ticket machine context.

4. Analysis of Combined Process Actions

- **Alphabets** of the combined processes, including the final process labelling.
- **Synchronous** actions are performed by at least two sub-process in the combination.
- **Blocked Synchronous** actions cannot be performed, because at least one of the sub-processes can never 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, e.g. if they include indexes in[0], in[1], …, in[5] as in[1..5]. Add rows as necessary.

Processes	**Alphabet** (Use LTSA's **compressed notation**, if alphabet is large.)
pas1.PASSENGER(1)	{exit, pas1.{acquireTicketMachine, printTicket, releaseTicketMachine}}
pas2.PASSENGER(1)	{exit, pas2.{acquireTicketMachine, printTicket, releaseTicketMachine}}
pas3.PASSENGER(1)	{exit, pas3.{acquireTicketMachine, printTicket, releaseTicketMachine}}
paperTech.PAPER_TECH	{exit, paperTech.{acquirePaperRefill, paperRefill, releasePaperRefill}}
tonerTech.TONER_TECH	{exit, tonerTech.{acquireTonerRefill, releaseTonerRefill, tonerRefill}}
TICKET_PRINTER	{paperTech.{acquirePaperRefill, paperRefill, releasePaperRefill}, {pas1, pas2, pas3}.{acquireTicketMachine, printTicket, releaseTicketMachine}, tonerTech.{acquireTonerRefill, releaseTonerRefill, tonerRefill}}

Synchronous Actions	**Synchronised by Sub-Processes (List)**
acquireTicketMachine	pas1, pas2, pas3, TICKET_PRINTER
printTicket	pas1, pas2, pas3, TICKET_PRINTER
releaseTicketMachine	pas1, pas2, pas3, TICKET_PRINTER
acquirePaperRefill	paperTech, pas1, pas2, pas3, TICKET_PRINTER
paperRefill	paperTech, pas1, pas2, pas3, TICKET_PRINTER
releasePaperRefill	paperTech, pas1, pas2, pas3, TICKET_PRINTER
acquireTonerRefill	tonerTech, pas1, pas2, pas3, TICKET_PRINTER

tonerRefill	tonerTech, pas1, pas2, pas3, TICKET_PRINTER
releaseTonerRefill	tonerTech, pas1, pas2, pas3, TICKET_PRINTER
exit	pas1, pas2, pas3, paperTech, tonerTech, TICKET_PRINTER

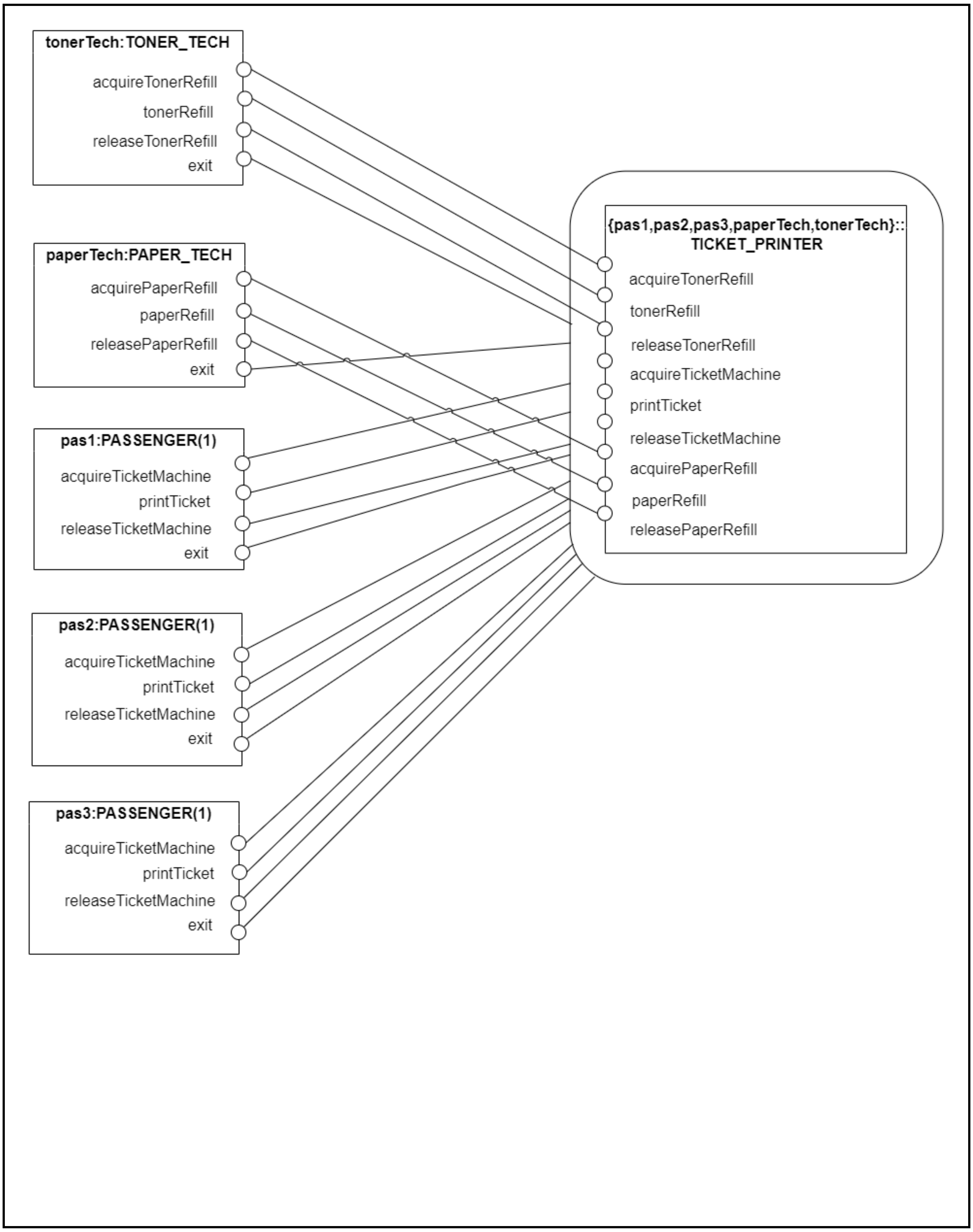
Blocked Synchronous Actions	**Blocking Processes**	**Blocked Processes**
PAPER_TECH.acquirePaperRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), tonerTech: TONER_TECH	paperTech: PAPER_TECH
PAPER_TECH.paperRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), tonerTech: TONER_TECH	paperTech: PAPER_TECH
PAPER_TECH.releasePaperRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), tonerTech: TONER_TECH	paperTech: PAPER_TECH
TICKET_PRINTER.acquireTicketMachine	paperTech: PAPER_TECH tonerTech: TONER_TECH	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1),
TICKET_PRINTER.printTicket	paperTech: PAPER_TECH tonerTech: TONER_TECH	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1),
TICKET_PRINTER.releaseTicketMachine	paperTech: PAPER_TECH tonerTech: TONER_TECH	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1),
TONER_TECH.acquireTonerRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), paperTech: PAPER_TECH	tonerTech: TONER_TECH

TONER_TECH.tonerRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), paperTech: PAPER_TECH	tonerTech: TONER_TECH
TONER_TECH.releaseTonerRefill	pas1: PASSENGER(1), pas2: PASSENGER(1), pas3: PASSENGER(1), paperTech: PAPER_TECH	tonerTech: TONER_TECH

Sub-Processes	**Asynchronous Actions (List)**
pas1.PASSENGER(1)	none
pas2.PASSENGER(1)	none
pas3.PASSENGER(1)	none
paperTech.PAPER_TECH	none
tonerTech.TONER_TECH	none
TICKET_PRINTER	none

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.


tonerTech:TONER_TECH
acquireTonerRefill
tonerRefill
releaseTonerRefill
exit
paperTech:PAPER_TECH
acquirePaperRefill
paperRefill
releasePaperRefill
exit
pas1:PASSENGER(1)
acquireTicketMachine
printTicket
releaseTicketMachine
exit
pas2:PASSENGER(1)
acquireTicketMachine
printTicket
releaseTicketMachine
exit
pas3:PASSENGER(1)
acquireTicketMachine
printTicket
releaseTicketMachine
exit
{pas1,pas2,pas3,paperTech,tonerTech}::
TICKET_PRINTER
acquireTonerRefill
tonerRefill
releaseTonerRefill
acquireTicketMachine
printTicket
releaseTicketMachine
acquirePaperRefill
paperRefill
releasePaperRefill