6SENG006W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Govinna Pathirathnage Sachin Lakshan
Student ID	UoW – **w1833545** | IIT - **20200603**
Date	02/01/2024

1. FSP Composition Process Attributes

Attribute	Value
Name	PURCHASE_TICKET_SYSTEM
Description	Models a ticket machine, two passengers, and two technicians—one for ticket paper and another for toner.
Sub-processes (List them.)	a:PASSENGER(3), b:PASSENGER(2), pt:TICKET_TECHNICIAN tt:TONER_TECHNICIAN {a,b,pt,tt}::TICKET_MACHINE
Number of States	62
Deadlocks (yes/no)	Yes (potential DEADLOCK)
Deadlock Trace(s) (If applicable)	a.acquireTicketMachine a.print a.releaseTicketMachine a.acquireTicketMachine a.print a.releaseTicketMachine a.acquireTicketMachine a.print a.releaseTicketMachine pt.acquireRefillPaper pt.refillPaper pt.releaseRefillPaper b.acquireTicketMachine b.print b.releaseTicketMachine tt.acquireRefillToner tt.refillToner tt.releaseRefillToner b.acquireTicketMachine b.print b.releaseTicketMachine 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 individual sub-processes.)

FSP Program:

```
const MAX_PAPERS = 3
const MAX_TONER = 4
set ACTIONS = {acquireTicketMachine, print, releaseTicketMachine,
                        acquireRefillPaper, refillPaper, releaseRefillPaper,
                        acquireRefillToner, refillToner, releaseRefillToner}

||PURCHASE_TICKET_SYSTEM = (a:PASSENGER(3) || b:PASSENGER(2)
                                || pt:TICKET_TECHNICIAN || tt:TONER_TECHNICIAN
                                || {a,b,pt,tt}::TICKET_MACHINE
)/{terminate/{a.terminate, b.terminate, pt.terminate, tt.terminate}}.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
a:PASSENGER(3)	Represents a passenger who uses the ticket machine to print a ticket.
b:PASSENGER(2)	Represents another passenger who uses the ticket machine to print a ticket.
pt:TICKET_TECHNICIAN	Represents a ticket paper technician refilling the machine when it's out of paper.
tt:TONER_TECHNICIAN	Represents a toner technician refilling the machine when it's out of toner.
{a,b,pt,tt}::TICKET_MACHINE	Represents a ticket machine that prints tickets for passengers and is refilled with paper and toner by relevant technicians. Therefore, this ticket machine acts as a shared resource between passengers and technicians.

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.)
a:PASSENGER(3)	{ a.acquireRefillPaper, a.acquireRefillToner, a.acquireTicketMachine, a.print, a.refillPaper, a.refillToner, a.releaseRefillPaper, a.releaseRefillToner, a.releaseTicketMachine, terminate }
b:PASSENGER(2)	{ b.acquireRefillPaper, b.acquireRefillToner, b.acquireTicketMachine, b.print, b.refillPaper, b.refillToner, b.releaseRefillPaper, b.releaseRefillToner, b.releaseTicketMachine, terminate }
pt:TICKET_TECHNICIAN	{ pt.acquireRefillPaper, pt.acquireRefillToner, pt.acquireTicketMachine, pt.print, pt.refillPaper, pt.refillToner, pt.releaseRefillPaper, pt.releaseRefillToner, pt.releaseTicketMachine }
tt:TONER_TECHNICIAN	{ tt.acquireRefillPaper, tt.acquireRefillToner, tt.acquireTicketMachine, tt.print, tt.refillPaper, tt.refillToner, tt.releaseRefillPaper, tt.releaseRefillToner, tt.releaseTicketMachine }
{a,b,pt,tt}::TICKET_MACHINE	{a, b, pt, tt}.{ acquireRefillPaper, acquireRefillToner, acquireTicketMachine, print, refillPaper, refillToner, releaseRefillPaper, releaseRefillToner, releaseTicketMachine }

Synchronous Actions	**Synchronised by Sub-Processes (List)**
a.acquireTicketMachine, a.print, a.releaseTicketMachine	a:PASSENGER(3), {a,b,pt,tt}::TICKET_MACHINE
b.acquireTicketMachine, b.print, b.releaseTicketMachine	b:PASSENGER(2), {a,b,pt,tt}::TICKET_MACHINE
pt.acquireRefillPaper, pt.refillPaper, pt.releaseRefillPaper	pt:TICKET_TECHNICIAN, {a,b,pt,tt}::TICKET_MACHINE

tt.acquireRefillToner, tt.refillToner, tt.releaseRefillToner	tt:TONER_TECHNICIAN, {a,b,pt,tt}::TICKET_MACHINE
terminate	a:PASSENGER(3), b:PASSENGER(2),

Blocked Synchronous Actions	Blocking Processes	Blocked Processes
a.acquireRefillPaper, a.acquireRefillToner, a.refillPaper, a.refillToner, a.releaseRefillPaper, a.releaseRefillToner	a:PASSENGER(3)	{a,b,pt,tt}::TICKET_MACHINE
b.acquireRefillPaper, b.acquireRefillToner, b.refillPaper, b.refillToner, b.releaseRefillPaper, b.releaseRefillToner	b:PASSENGER(2),	{a,b,pt,tt}::TICKET_MACHINE
pt.acquireTicketMachine, pt.acquireRefillToner, pt.print, pt.refillToner, pt.releaseTicketMachine, pt.releaseRefillToner,	pt:TICKET_TECHNICIAN	{a,b,pt,tt}::TICKET_MACHINE
tt.acquireTicketMachine, tt. acquireRefillPaper, tt.print, tt.refillPaper, tt.releaseTicketMachine, tt.releaseRefillPaper,	tt:TONER_TECHNICIAN	{a,b,pt,tt}::TICKET_MACHINE

Sub-Processes	Asynchronous Actions (List)
a:PASSENGER(3)	None
b:PASSENGER(2)	None
pt:TICKET_TECHNICIAN	None
tt:TONER_TECHNICIAN	None
{a,b,pt,tt}::TICKET_MACHINE	None

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.