6SENG006W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Achintha Jayatilake
Student ID	2019530
Date	08/01/2024

1. FSP Composition Process Attributes

Attribute	Value
Name	PRINTING _SYSTEM
Description	A system that manages ticket printing with two passengers and a technician, taking into account paper and toner levels in the printer.
Sub-processes (List them.)	passenger1:PASSENGER(3) , passenger2:PASSENGER(2) , technician : TECHNICIAN :ADVANCED_TICKETING_SYSTEM
Number of States	43736
Deadlocks (yes/no)	No
Deadlock Trace(s) (If applicable)	Not applicalbe

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:

```
range PAPER_TRAY = 0..3
const EMPTY_PAPER_TRAY = 0
const FULL_PAPER_TRAY = 3

range TONER_LEVEL = 0..2
const EMPTY_TONER_LEVEL = 0
const FULL_TONER_LEVEL = 2

set Passengers = { passenger1, passenger2 }
set Technician = {paper_technician, torner_technician}
set Users = {Passengers, Technician}



set PassengerProhibitedActions = { acquireReloading, load, unableToload,
acquireTonerRefill}
set TechnicianProhibitedActions = {acquirePrinting}

||PRINTING_SYSTEM = (passenger1: PASSENGER(3) || passenger2: PASSENGER(2) ||
technician: TECHNICIAN || Users :: ADVANCED_TICKETING_SYSTEM) / {end/
Users.end}.
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
passenger1 : PASSENGER(3)	Represents a passenger in the ticketing system, likely with a set of actions they can perform related to ticket printing. The number (3) might indicate a priority or a limit on the number of actions they can perform.
passenger2 : PASSENGER(2)	Similar to passenger1, this represents another passenger with possibly different priorities or limits (indicated by the number 2).
technician	Represents a technician who is responsible for maintaining the printer. This role may include actions such as refilling paper and toner, and handling printer malfunctions.

PRINTER	This is the core printer process in the system. It should include the logic for printing tickets, managing paper and toner levels, and handling interactions with both passengers and the technician.

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.)
passenger1	{end, passenger1.{{acquirePrinting, acquireReloading, acquireTonerRefill, free, load}, printTicket[1..3], unableToload
passenger2	{end, passenger2.{{acquirePrinting, acquireReloading, acquireTonerRefill, free, load}, printTicket[1..2], unableToload}}
technician	{acquireReloading, reloadPaper, acquireTonerRefill, refillToner}
ADVANCED_PRINTER	technician.{acquirePrinting, acquireReloading, acquireTonerRefill, end, free, load, unableToload}

Synchronous Actions	**Synchronised by Sub-Processes (List)**
passenger1.{ acquirePrintings, acquireReloading, acquireTonerRefill, unableToLoad, load, free }	passenger1: PASSENGER(3), ADVANCED_PRINTER
passenger2.{ acquirePrintings, acquireReloading, acquireTonerRefill, unableToLoad, load, free }	passenger2: PASSENGER(2), ADVANCED_PRINTER
technician.{acquireReloading, reloadPaper, acquireTonerRefill, refillToner}	technician: TECHNICIAN, ADVANCED_PRINTER
end	passenger1: PASSENGER(3), passenger2: PASSENGER(2), technician: TECHNICIAN

Blocked Synchronous Actions	**Blocking Processes**	**Blocked Processes**
technician.{acquirePrintings}	technician: TECHNICIAN, ADVANCED_PRINTER	technician: TECHNICIAN

passenger1{ acquireReloading, load, unableToload, acquireTonerRefill}	passenger1: PASSENGER(3), ADVANCED_PRINTER	passenger1: PASSENGER(3)
passenger2{ acquireReloading, load, unableToload, acquireTonerRefill}	passenger2: PASSENGER(2), ADVANCED_PRINTER	passenger2: PASSENGER(2)

Sub-Processes	Asynchronous Actions (List)
passenger1 : PASSENGER(3)	passenger1.requestTicket, passenger1.collectTicket
passenger2 : PASSENGER(2)	passenger2.requestTicket, passenger2.collectTicket
ADVANCED_PRINTER	None (All actions are synchronous with passengers and technician)
technician : TECHNICIAN	technician.detectIssue, technician.fixIssue

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.