6SENG006C.1 Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Undupitiya Appulage Pragathi Sithmini
Student ID	20200351
Date	January 2024

1. FSP Composition Process Attributes

Attribute	Value
Name	PURCHASE_TICKET_SYSTEM
Description	This process consists of a ticket printer which is used by passenger and technicians.
Alphabet (Use LTSA's compressed notation, if alphabet is large.)	firstPassenger.(accessRefillPaperOption, accessRefillTonerOption, issueTicket, obtainMachine, refillPaper, refillToner, release, releaseRefillPaper, releaseRefillToner) secondPassenger.(accessRefillPaperOption, accessRefillTonerOption, issueTicket, obtainMachine, refillPaper,.refillToner, release, releaseRefillPaper, releaseRefillToner) ticketTechnician.(accessRefillPaperOption, accessRefillTonerOption, issueTicket, obtainMachine, refillPaper, refillToner, releaseRefillPaper, releaseRefillToner) tonerTechnician.(accessRefillPaperOption, accessRefillTonerOption, issueTicket, obtainMachine, refillPaper, refillToner, release, releaseRefillPaper, releaseRefillToner)
Sub-processes (List them.)	TICKET_MACHINE, PASSENGER, TICKET_TECHNICIAN, TONER_TECHNICIAN
Number of States	15
Deadlocks (yes/no)	No deadlocks/errors
Deadlock Trace(s) (If applicable)	N/A

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 MAXIMUM_PAPER = 3
range PAPER_RANGE = 0..MAXIMUM_PAPER

const MAXIMUM_TONER = 3
range TONER_RANGE = 0..MAXIMUM_TONER

// Define a set of machine actions
set MACHINE_ACTIONS = {obtainMachine, issueTicket, release, accessRefillTonerOption,
refillToner, releaseRefillToner, accessRefillPaperOption, refillPaper, releaseRefillPaper}

|| PURCHASE_TICKET_SYSTEM =
  (firstPassenger: PASSENGER(3)
   || secondPassenger: PASSENGER(2)
   || tonerTechnician: TONER_TECHNICIAN
   || ticketTechnician: TICKET_TECHNICIAN
   || {firstPassenger, secondPassenger, tonerTechnician, ticketTechnician} :: TICKET_MACHINE)
  / {terminate / {firstPassenger.terminate, secondPassenger.terminate, tonerTechnician.terminate,
ticketTechnician.terminate}} .
```

3. Combined Sub-processes

Process	Description
TICKET_MACHINE	Represents a ticket printer machine can print ticket.
PASSENGER	Represents a passenger who needs to print a ticket.
TICKET_TECHNICIAN	Represents a paper(ticket) technician who refills paper whenever the ticket printer is out of papers.
TONER_TECHNICIAN	Represents a toner technician who refills toner whenever the ticket printer is out of toner.

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. \

Synchronous Actions	Synchronised by Sub-Processes (List)
obtainMachine, issueTicket, release	TICKET_MACHINE, PASSENGER
accessRefillPaperOption, refillPaper, releaseRefillPaper	TICKET_MACHINE, TICKET_TECHNICIAN
accessRefillTonerOption, refillToner, releaseRefillToner	TICEKT_MACHINE, TONER_TECHINICIAN

Sub-Process	Asynchronous Actions (List)
PASSENGER	terminate, END

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.