6SENG006W Concurrent Programming

FSP Process Analysis & Design Form

Name	Sam Clark
Student ID	W1854525
Date	05/01/2024

1. FSP Process Attributes

Attribute	Value
Name	PURCHASE_TICKET_SYSTEM
Description	Represents a ticket machine that prints passenger tickets and requires paper & toner refills when empty.
Alphabet	checkPaper, checkToner, p.acquire, p.refillPaper, p.release, p.wait, ps.acquire, ps.finish, ps.printTicket, ps.release, refillPaper, refillToner, remainingTickets, t.acquire, t.refillToner, t.wait t.release
Number of States	3104
Deadlocks (yes/no)	No
Deadlock Trace(s) (if applicable)	N/A

2. FSP Process Code

FSP Process:

```
const MAX_PAPER = 5
const MAX_TONER = 5

range PAPER_CAPACITY = 0 .. MAX_PAPER
range TONER_CAPACITY = 0 .. MAX_TONER

const MIN_PRINT = 1
const MAX_PRINT = 8

range TICKETS_TO_PRINT = MIN_PRINT .. MAX_PRINT

PAPER_TECH(INITIAL_PAPER = 3) = Paper[INITIAL_PAPER] ,

Paper[currentPaper : PAPER_CAPACITY] = (when(currentPaper == 0) p.acquire -> refillPaper
-> p.release -> Paper[MAX_PAPER]
                                                    | when(currentPaper > 0)
p.acquire -> p.wait -> p.release -> Paper[currentPaper - 1]).

TONER_TECH(INITIAL_TONER = MAX_TONER) = Toner[INITIAL_TONER] ,

Toner[currentToner : TONER_CAPACITY] = (checkToner[currentToner] ->
Toner[currentToner]
                                                        | when(currentToner ==
0) t.acquire -> refillToner -> t.release -> Toner[MAX_TONER]
                                                        | when(currentToner >
0) t.acquire -> t.release -> Toner[currentToner - 1]).



PASSENGER(TICKETS = MAX_PRINT) = Passenger[TICKETS] ,
```

```
Passenger[currentTickets : TICKETS_TO_PRINT] =      (remainingTickets[currentTickets] ->
Passenger[currentTickets]
                                                                                    |
when(currentTickets > 1) ps.acquire -> ps.printTicket -> ps.release ->
Passenger[currentTickets - 1]
                                                                                    |
when(currentTickets == 1) ps.acquire -> ps.printTicket -> ps.release -> ps.finish -> END).

PAPER_MACHINE(INITIAL_PAPER = 3) = Paper_Machine[INITIAL_PAPER] ,

Paper_Machine[currentPaper : PAPER_CAPACITY] = (checkPaper[currentPaper] ->
Paper_Machine[currentPaper]
                                                                    | when(currentPaper >
0) ps.acquire -> ps.printTicket -> ps.release -> Paper_Machine[currentPaper - 1]
                                                                    | when(currentPaper ==
0) p.acquire -> p.wait -> p.refillPaper -> p.release -> Paper_Machine[MAX_PAPER]
                                                                    | when(currentPaper >
0) p.acquire -> p.wait -> p.release -> Paper_Machine[currentPaper]).

TONER_MACHINE(INITIAL_TONER = 5) = Toner_Machine[INITIAL_TONER] ,

Toner_Machine[currentToner : TONER_CAPACITY] = (checkToner[currentToner] ->
Toner_Machine[currentToner]
                                                                    | when
(currentToner > 0) ps.acquire -> ps.printTicket -> ps.release -> Toner_Machine[currentToner
- 1]
                                                                    | when
(currentToner == 0) t.acquire -> t.refillToner -> t.release -> Toner_Machine[MAX_TONER]
                                                                    | when
(currentToner > 0) t.acquire -> t.wait -> t.release -> Toner_Machine[currentToner]).

||TICKET_MACHINE(INITIAL_PAPER = 3, INITIAL_TONER = 5) = (PAPER_MACHINE ||
```

TONER_MACHINE).

||PURCHASE_TICKET_SYSTEM = (PASSENGER || PAPER_TECH || TONER_TECH || TICKET_MACHINE) .

3. Actions Description

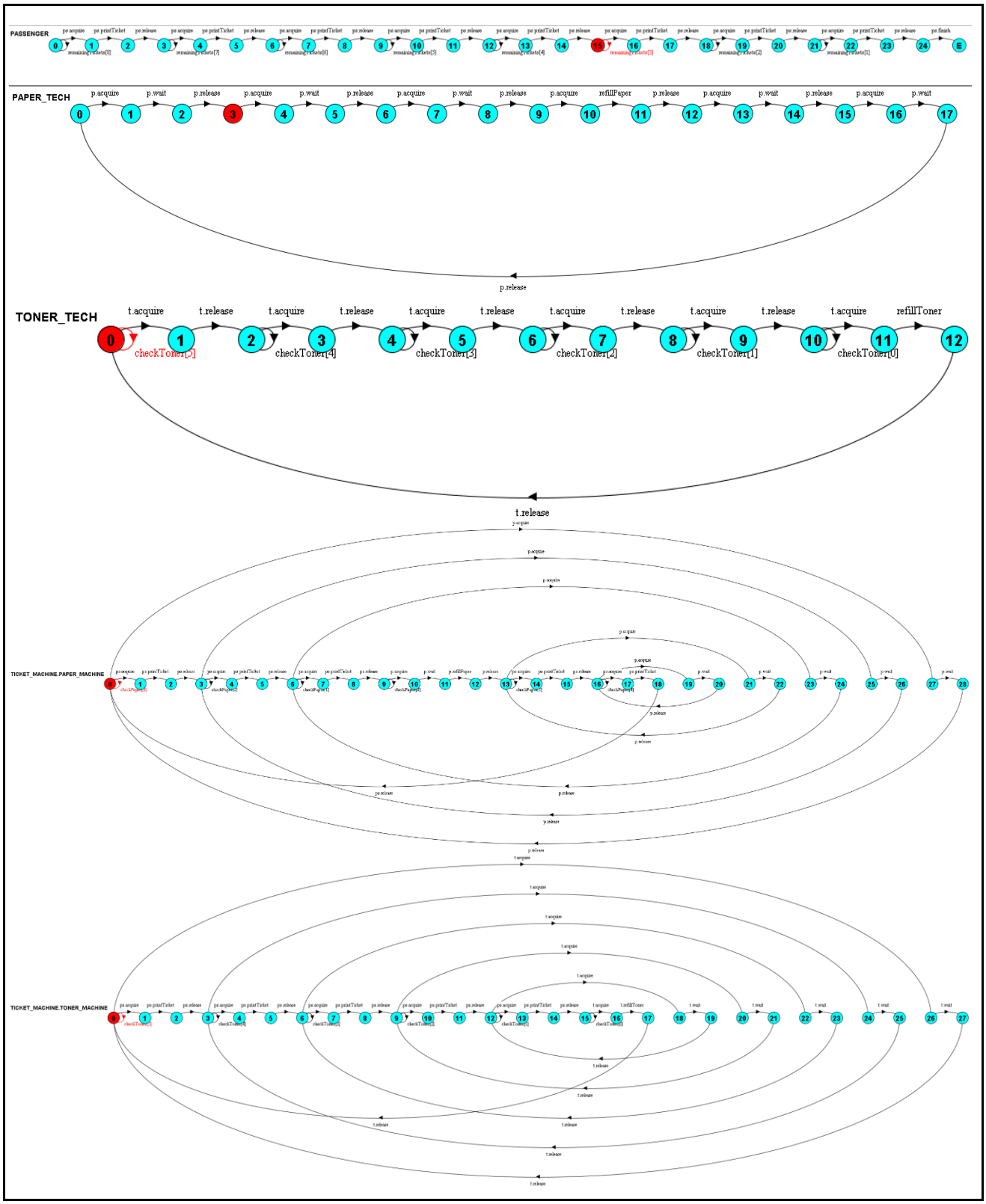
A description of what each of the FSP process' actions represents, i.e. is modelling. In addition, indicate if the action is intended to be synchronised (shared) with another process or asynchronous (not shared). (Add rows as necessary.)

Actions	Represents	Synchronous or Asynchronous
checkPaper	Represents the current level of paper in the machine	Synchronous
checkToner	Represents the current level of toner in the machine	Synchronous
remainingTickets	Represents the remaining tickets to be printed by the passenger	Synchronous
p.acquire	Paper technician gaining mutual exclusive access to the ticket machine	Asynchronous
p.refillPaper	Paper technician refilling the paper levels to maximum (5)	Asynchronous
p.wait	Paper technician waiting to perform action	Asynchronous
p.release	Paper technician releasing mutual exclusive access to the ticket machine	Asynchronous
t.acquire	Toner technician gaining mutual exclusive access to the ticket machine	Asynchronous
t.refillToner	Toner technician refilling toner levels to maximum (5)	Asynchronous
t.wait	Toner technician waiting to perform action	Asynchronous
t.release	Toner technician releasing mutual exclusive access to the ticket machine	Asynchronous
ps.acquire	Passenger gaining mutual exclusive access to the ticket machine	Asynchronous
ps.printTicket	Passenger prints one of their tickets and the	Asynchronous

	remainingTickets will be reduced by 1	
ps.release	Passenger releasing mutual exclusive access to the ticket machine	Asynchronous
ps.finish	Passenger finishes printing their tickets, no remaining tickets to print so passenger ends its actions	Asynchronous

4. FSM/LTS Diagrams of FSP Process

Note that if there are too many states, more than 64, then the LTSA tool will not be able to draw the diagram. In this case draw small diagrams of the most important parts of the complete diagram.



There were too many states (3104) for the complete purchase ticket system, so I have included each of the processes that make up the larger combined machine.

5. LTS States

A description of what each of the FSP process' states represents, i.e. is modelling. If there are a large number of states then you can group similar states together &/or only include the most important ones. For example, identify any states related to mutual exclusion (ME) & the associated critical section (CS), e.g. waiting to enter the CS state, in the CS state(s), left the CS state. (Add rows as necessary.)

State	Represents
Too Many States	Ticket machine is waiting for passenger to start printing tickets
Too Many States	The passenger has started using the machine, acquiring exclusive access to it
Too Many States	The passenger prints one of their tickets, reducing the number of tickets they still need to print by 1
Too Many States	The passenger releases exclusive access to the ticket machine temporarily before they print their next ticket
Too Many States	The paper technician gains exclusive access to the ticket machine to perform refills
Too Many States	The paper technician refills the paper levels back to maximum, ready for more printing
Too Many States	The paper technician releases exclusive access to the ticket machine
Too Many States	The toner technician gains exclusive access to the ticket machine
Too Many States	The toner technician refills the toner levels back to maximum, ready for more printing
Too Many States	The toner technician releases exclusive access to the ticket machine
Too Many States	Passenger finishes printing their tickets and ends their use of the machine
	Note: Due to the ticket machine being too large in terms of states, I cannot see the individual states on which each of these events occur, however I have included the possible states that the program can be in.

6. Trace Tree for FSP Process

The trace tree for the process. Use the conventions given in the lecture notes and add explanatory notes if necessary.

Too many states to define a trace tree.