**6SENG006W Concurrent Programming**

**FSP Process Composition Analysis & Design Form**

|  |  |
| --- | --- |
| **Name** | Akila Edirisooriya |
| **Student ID** | 2019037/ W1809738 |
| **Date** | 1/11/2024 |

**1. FSP Composition Process Attributes**

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| **Name** | PURCHASE\_TICKET\_SYSTEM |
| **Description** | This is a process of a ticket machine, which can print tickets, refill paper, and print tickets. The passengers, who can print tickets and terminate; and the technicians, who can refill paper or toner. |
| **Sub-processes**  (List them.) | {{a, b, p, t}.{acquirePrinter, acquireRefill, acquireTonerRefill, print, refill, refillToner, releasePrinter, releaseRefill, releaseTonerRefill, start}, terminate} |
| **Number of States** | 66 |
| **Deadlocks**  (yes/no) | NO |
| **Deadlock Trace(s)**  **(If applicable)** | NONE |

**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\_TICKET = 3  set ACTIONS ={acquirePrinter,print, releasePrinter ,acquireRefill,refill,releaseRefill,releaseTonerRefill,refillToner,acquireTonerRefill}  // The printer process  TICKET\_MACHINE = (start -> TICKET\_MACHINE[MAX\_TICKET]),  TICKET\_MACHINE [i:0..MAX\_TICKET] = if (i > 0) then  (acquirePrinter -> print -> releasePrinter -> TICKET\_MACHINE[i-1])  else  (acquireRefill -> refill -> releaseRefill-> acquireTonerRefill -> refillToner -> releaseTonerRefill -> TICKET\_MACHINE[MAX\_TICKET]).  // The user process  PASSENGER (COUNT = MAX\_TICKET ) = PASSENGER[COUNT],  PASSENGER [i:0..COUNT] = if (i > 0) then  (acquirePrinter -> print -> releasePrinter -> PASSENGER[i-1])  else  (terminate -> END)+ ACTIONS.  // The paper technician process  PAPERTECHNICIAN = (acquireRefill -> refill -> releaseRefill -> PAPERTECHNICIAN |terminate -> END)+ ACTIONS.  // The toner technician process  TONERTECHNICIAN = (acquireTonerRefill -> refillToner -> releaseTonerRefill -> TONERTECHNICIAN | terminate -> END) + ACTIONS.  // The parallel composition of the system  ||PURCHASE\_TICKET\_SYSTEM = ({a,b,p,t}::TICKET\_MACHINE || a:PASSENGER(3)||b:PASSENGER(2) || p:PAPERTECHNICIAN||t:TONERTECHNICIAN)  /{terminate/{a.terminate, b.terminate, p.terminate,t.terminate}}. |

**3. Combined Sub-processes**

(Add rows as necessary.)

|  |  |
| --- | --- |
| **Process** | **Description** |
| PASSENGER(3) | This passenger is having three tickets to print using the printer. |
| PASSENGER(2) | This passenger is having four tickets to print using the printer. |
| PAPERTECHNICIAN | The technician refills the printer with the papers when the printer run outs of  paper. When the printer is out of papers  The paper technician refills the printer with the papers when the printer run outs of paper. When the printer is out of papers |
| TONERTECHNICIAN | The toner technician refills the printer with the toner when the printer run outs of toner. When the printer is out of papers |
|  |  |
|  |  |
|  |  |
|  |  |

**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.) |
| PASSENGER(3) | {a.{acquirePrinter, acquireRefill, acquireTonerRefill, print, refill, refillToner, releasePrinter, releaseRefill, releaseTonerRefill}, terminate} |
| PASSENGER(2) | {b.{acquirePrinter, acquireRefill, acquireTonerRefill, print, refill, refillToner, releasePrinter, releaseRefill, releaseTonerRefill}, terminate} |
| PAPERTECHNICIAN | {p.{acquirePrinter, acquireRefill, acquireTonerRefill, print, refill, refillToner, releasePrinter, releaseRefill, releaseTonerRefill}, terminate} |
| TONERTECHNICIAN | {t.{acquirePrinter, acquireRefill, acquireTonerRefill, print, refill, refillToner, releasePrinter, releaseRefill, releaseTonerRefill}, terminate} |

|  |  |
| --- | --- |
| **Synchronous Actions** | **Synchronised by Sub-Processes (List)** |
| passenge1.acquire  passenger1.print[1]  passenger1.print[2]  passenger1.print[3]  passenger1.release | PASSENGER (3), PRINTER |
| passenge2.acquire  passenge2.print[1]  passenge2.print[2]  passenge2.release | PASSENGER (2), PRINTER |
| technician.acquireTech  technician.refillPrinter  technician.release  papertechnician. acquirePrinter  papertechnician.refill  papertechnician.releasePrinter | PAPERTECHNICIAN, PRINTER |
| tonertechnician. acquirePrinter  tonertechnician. refill  tonertechnician. releasePrinter | TONERTECHNICIAN, PRINTER |
| outOfPaperAlert | PAPERTECHNICIAN, PRINTER |
| outOfTonerAlert | TONERTECHNICIAN, PRINTER |
| terminate | PASSENGER (3), PASSENGER (2), PAPERTECHNICIAN, TONERTECHNICIAN |

|  |  |  |
| --- | --- | --- |
| **Blocked**  **Synchronous Actions** | **Blocking Processes** | **Blocked Processes** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Sub-Processes** | **Asynchronous Actions (List)** |
| PASSENGER (3) | None |
| PASSENGER (2) | None |
| PAPERTECHNICIAN | papertechnician. acquirePrinter  papertechnician.refill |
| TONERTECHNICIAN | tonertechnician. acquirePrinter  tonertechnician. refill |
| PRINTER | None |

**5. Parallel Composition Structure Diagram**

The structure diagram for the parallel composition.

|  |
| --- |
|  |