**6SENG006W Concurrent Programming**

**FSP Process Analysis & Design Form**

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

**1. FSP 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. |
| **Alphabet** | {{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 Process Code**

|  |
| --- |
| **FSP Process:** |
| 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. 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** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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.

|  |
| --- |
|  |

**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** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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.

|  |
| --- |
|  |