# 6SENG002W Concurrent Programming

# FSP Process Analysis & Design Form

| | |
|---|---|
| **Name** | K.K.Resindu Navoda |
| **Student ID** | 2016352/w1654203 |
| **Date** | 2020/01/06 |

## 1. FSP Process Attributes

| Attribute | Value |
|---|---|
| **Name** | STUDENT |
| **Description** | Money withdrawal process from a bank account by a student and buy a smart phone. |
| **Alphabet** | { readCurrentBalance, subtractMoney, buyPhone, calculateNewBalance, updateNewBalance, validateNewBalance} |
| **Number of States** | 6 |
| **Deadlocks (yes/no)** | No |
| **Deadlock Trace(s)** | No |

## 2. FSP Process Code

**FSP Process:**

```
STUDENT    =(readCurrentBalance->    WITHDRAW),
WITHDRAW=  (subtractMoney->      CALCULATE_BALANCE),
CALCULATE_BALANCE=   (calculateNewBalance->    BUY_PHONE),
BUY_PHONE=(buyPhone  ->    updateNewBalance -> validateNewBalance->
      STUDENT).
```

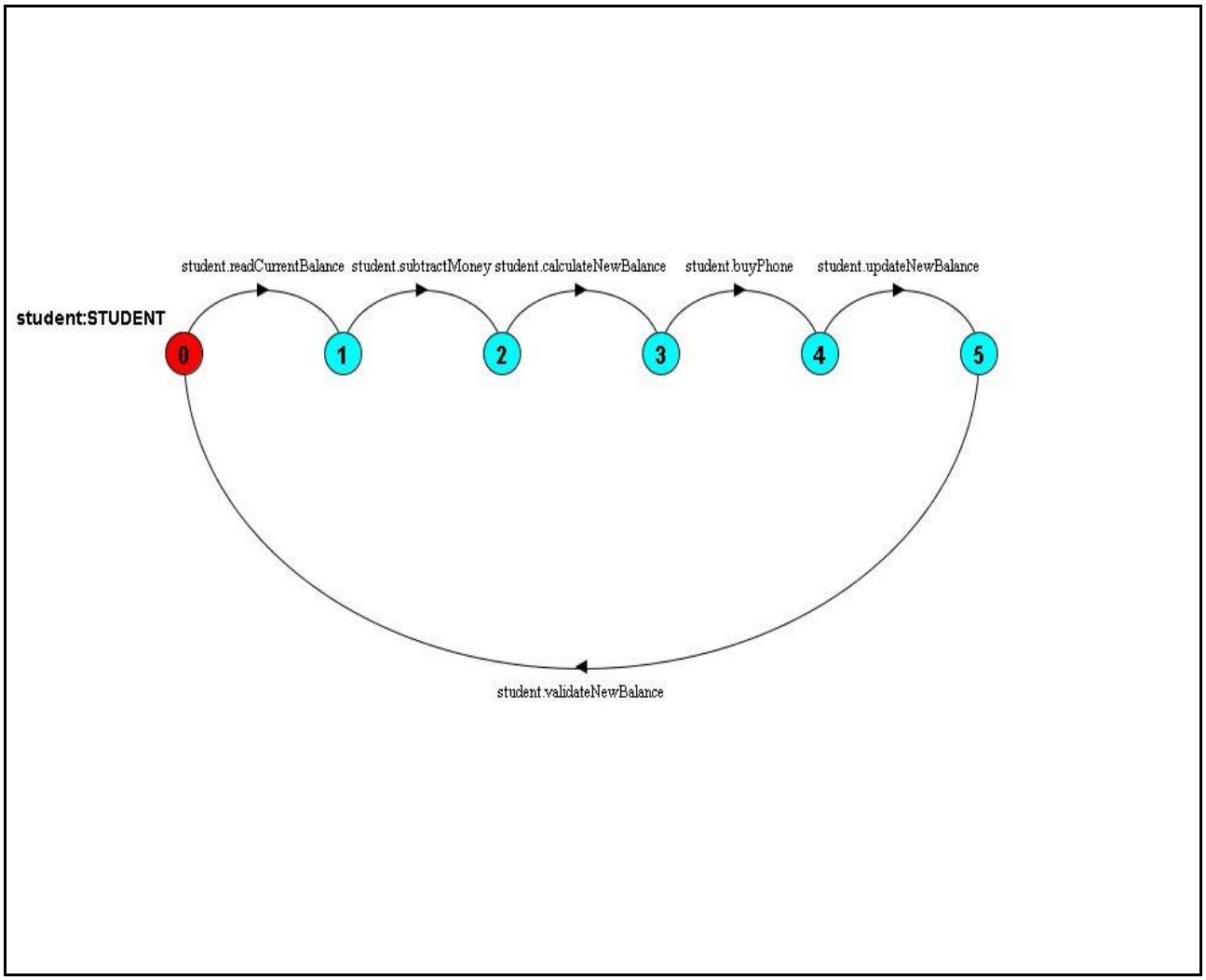
## 3. Actions Description

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 |
|---|---|---|
| readCurrentBalance | Student reads the current bank account balance | Synchronized |
| subtractMoney | Reduce the withdrawal money amount from the bank account | Asynchronized |
| buyPhone | Student buying a smart phone | Synchronized |
| calculateNewBalance | Calculate the bank account balance after the withdrawal of money | Asynchronized |
| updateNewBalance | Update the bank account balance after the withdrawal of money | Synchronized |
| validateNewBalance | Validate the updated bank account balance | Synchronized |
| | | |
| | | |
| | | |

## 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.



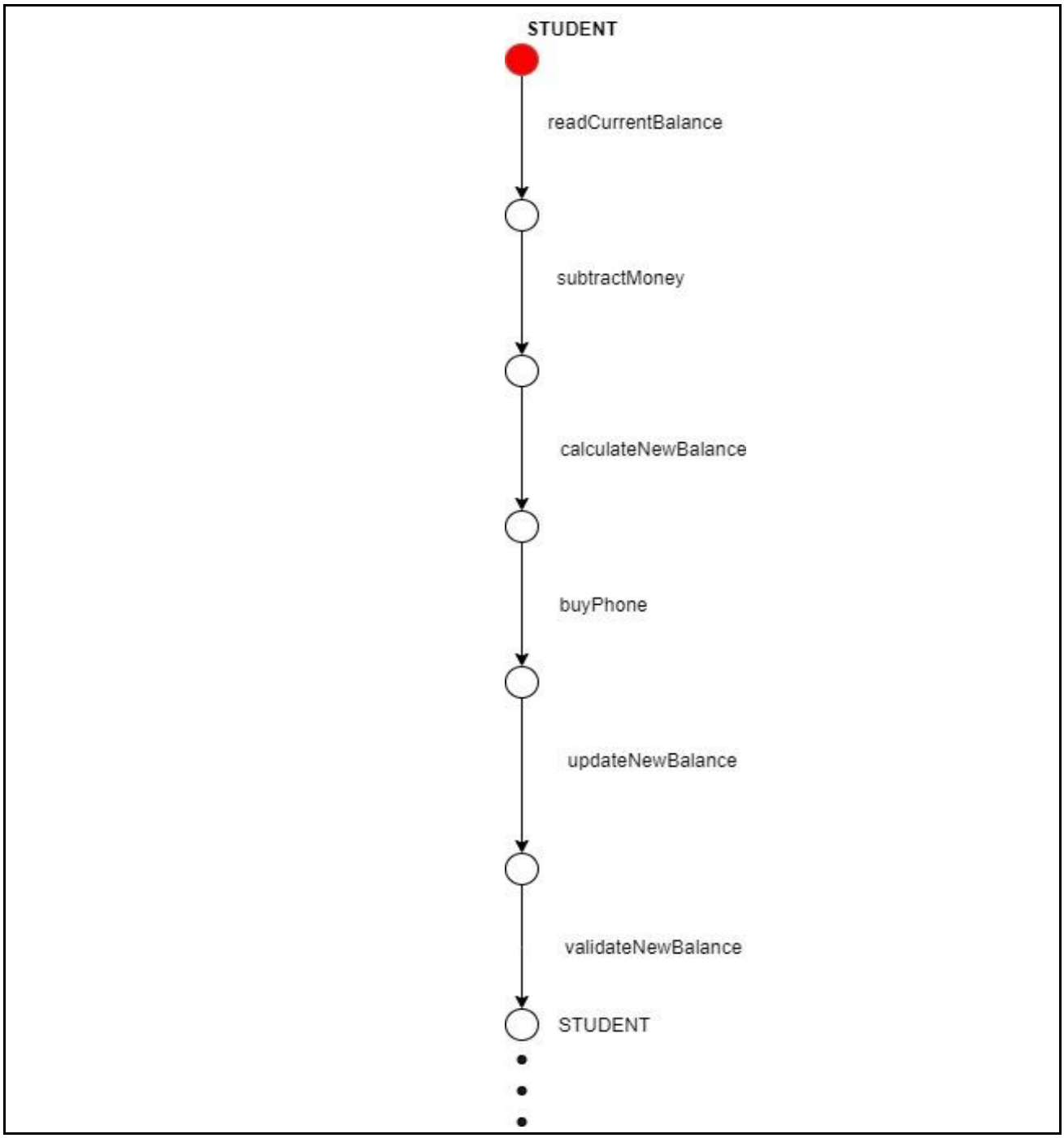
## 5. LTS States

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.)

| States | Represents |
|---|---|
| 0 | Student reads his/her own bank account balance |
| 1 | withdrawal money amount from the bank account balance is subtracted from the balance |
| 2 | The bank account balance after the withdrawal of money is calculated |
| 3 | Student buying smart phone |
| 4 | Bank account balance after the withdrawal of money is updated |
| 5 | Updated bank account balance after the withdrawal is validated. |
| | |
| | |
| | |
| | |
| | |
| | |

## 6. Trace Tree for FSP Process

The trace tree for the process. Use the conventions given in the lecture notes.



## 1. FSP Process Attributes

| Attribute | Value |
|---|---|
| **Name** | GRANDMOTHER |
| **Description** | Money deposit process to the student's bank account by a his/her grandmother as a birthday present and send e birthday card. |
| **Alphabet** | { readCurrentBalance, addCurrentBalance,calculateNewBalance, updateNewBalance, sendCard, validateNewBalance} |
| **Number of States** | |
| **Deadlocks (yes/no)** | No |
| **Deadlock Trace(s)** | No |

## 2. FSP Process Code

**FSP Process:**

```
GRANDMOTHER =(readCurrentBalance ->DEPOSIT),
DEPOSIT=  (addCurrentBalance  ->  CALCULATE_BALANCE),
CALCULATE_BALANCE=  (calculateNewBalance ->  SEND_CARD),
SEND_CARD =    (updateNewBalance   ->   sendCard  ->
     validateNewBalance  ->   GRANDMOTHER).
```

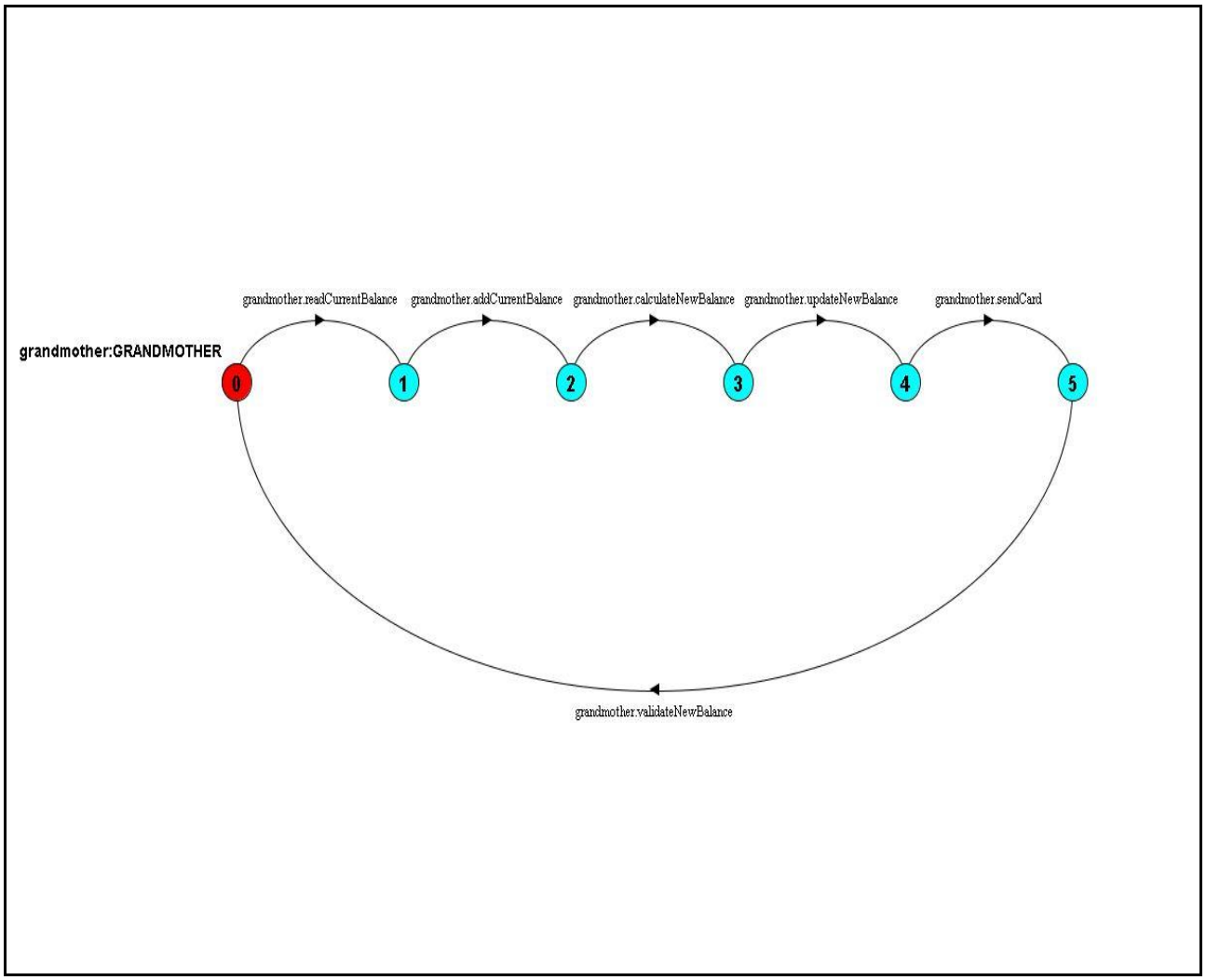
## 3. Actions Description

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 |
|---|---|---|
| readCurrentBalance | Grandmother reads the current bank account balance | Synchronized |
| addCurrentBalance | Add deposit money amount to the bank account | Asynchronized |
| calculateNewBalance | Calculate the bank account balance after depositing the money | Synchronized |
| updateNewBalance | Update the bank account balance after depositing the money | Synchronized |
| sendCard | Send a birthday card after grandmother deposit the money | Asynchronized |
| validateNewBalance | Validate the updated bank account balance | Synchronized |
| | | |
| | | |
| | | |

## 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.



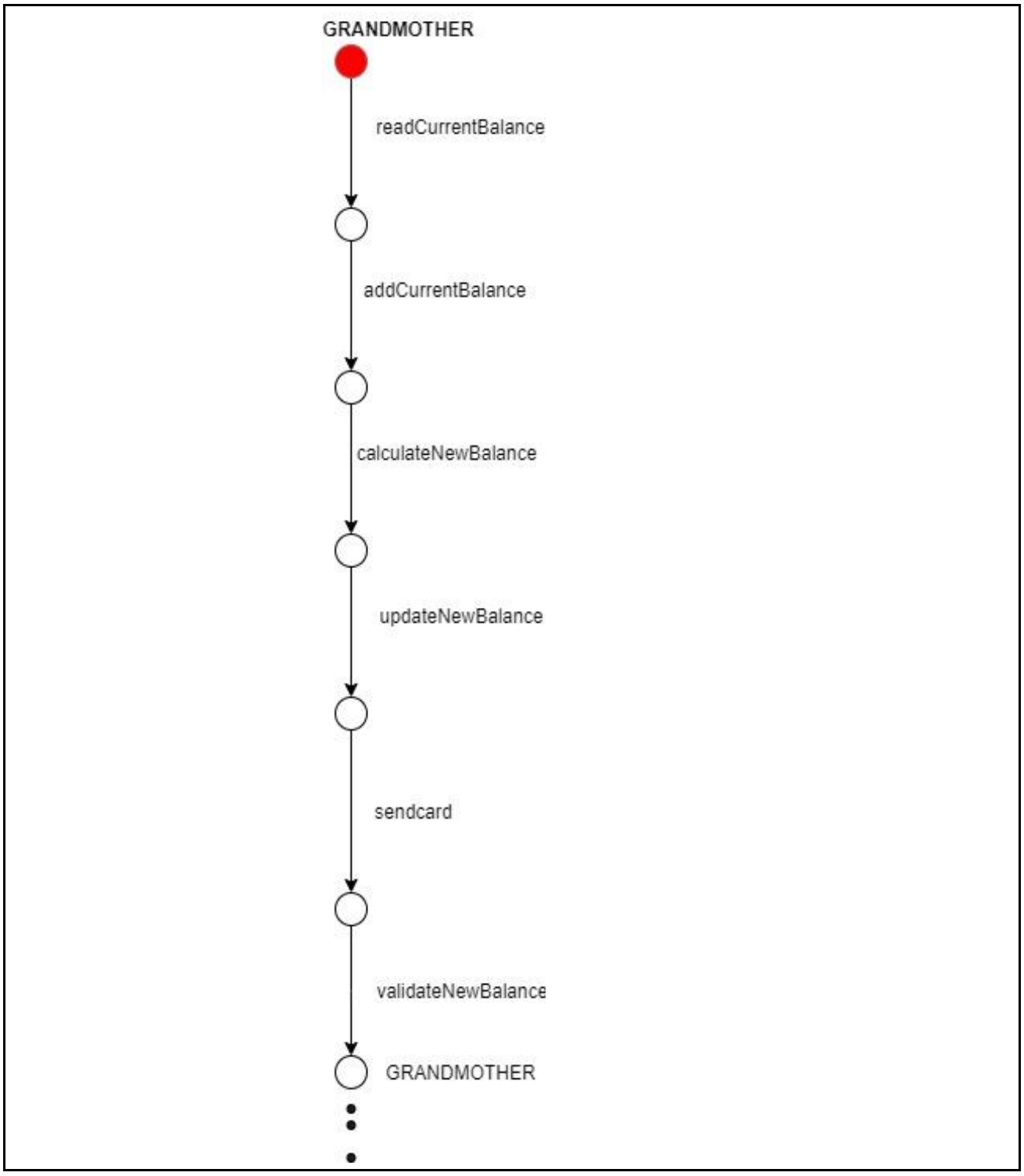
## 5. LTS States

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.)

| States | Represents |
|---|---|
| 0 | Grandmother reads student's bank account balance |
| 1 | Add the deposited amount of money to the bank account balance |
| 2 | The bank account balance after depositing the money is calculated |
| 3 | Bank account balance after depositing the money is updated |
| 4 | Sends a card |
| 5 | Updated bank account balance after the deposition is validated. |
| | |
| | |
| | |
| | |
| | |
| | |

## 6. Trace Tree for FSP Process

The trace tree for the process. Use the conventions given in the lecture notes.



## 1. FSP Process Attributes

| Attribute | Value |
|---|---|
| **Name** | UNIVERSITY |
| **Description** | Money withdrawal process from the bank account by the University |
| **Alphabet** | { readCurrentBalance, subtractMoney,calculateNewBalance, updateNewBalance, validateNewBalance} |
| **Number of States** | |
| **Deadlocks (yes/no)** | No |
| **Deadlock Trace(s)** | No |

## 2. FSP Process Code

**FSP Process:**

```
UNIVERSITY=     (readCurrentBalance->     subtractMoney->
      CALCULATE_BALANCE),
CALCULATE_BALANCE=   (calculateNewBalance->   updateNewBalance     ->
      validateNewBalance   ->   UNIVERSITY).
```

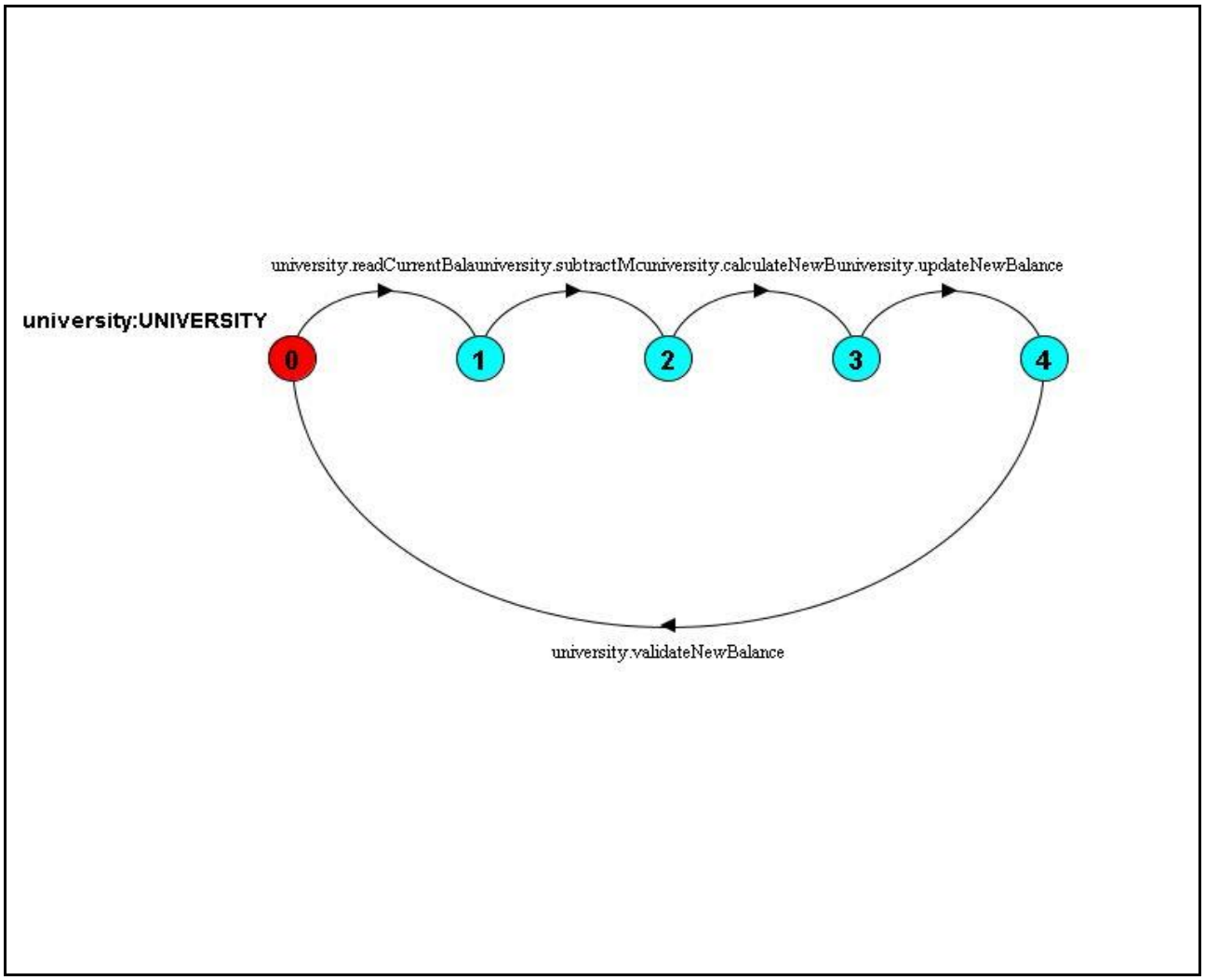
## 3. Actions Description

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 |
|---|---|---|
| readCurrentBalance | University reads the current bank account balance | Synchronous |
| subtractMoney | Reduce the withdrawal money amount from the bank account | Asynchronous |
| calculateNewBalance | Calculate the bank account balance after the withdrawal of money | Synchronous |
| updateNewBalance | Update the bank account balance after the withdrawal of money | Synchronous |
| validateNewBalance | Validate the updated bank account balance | Synchronous |
| | | |
| | | |
| | | |

## 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.



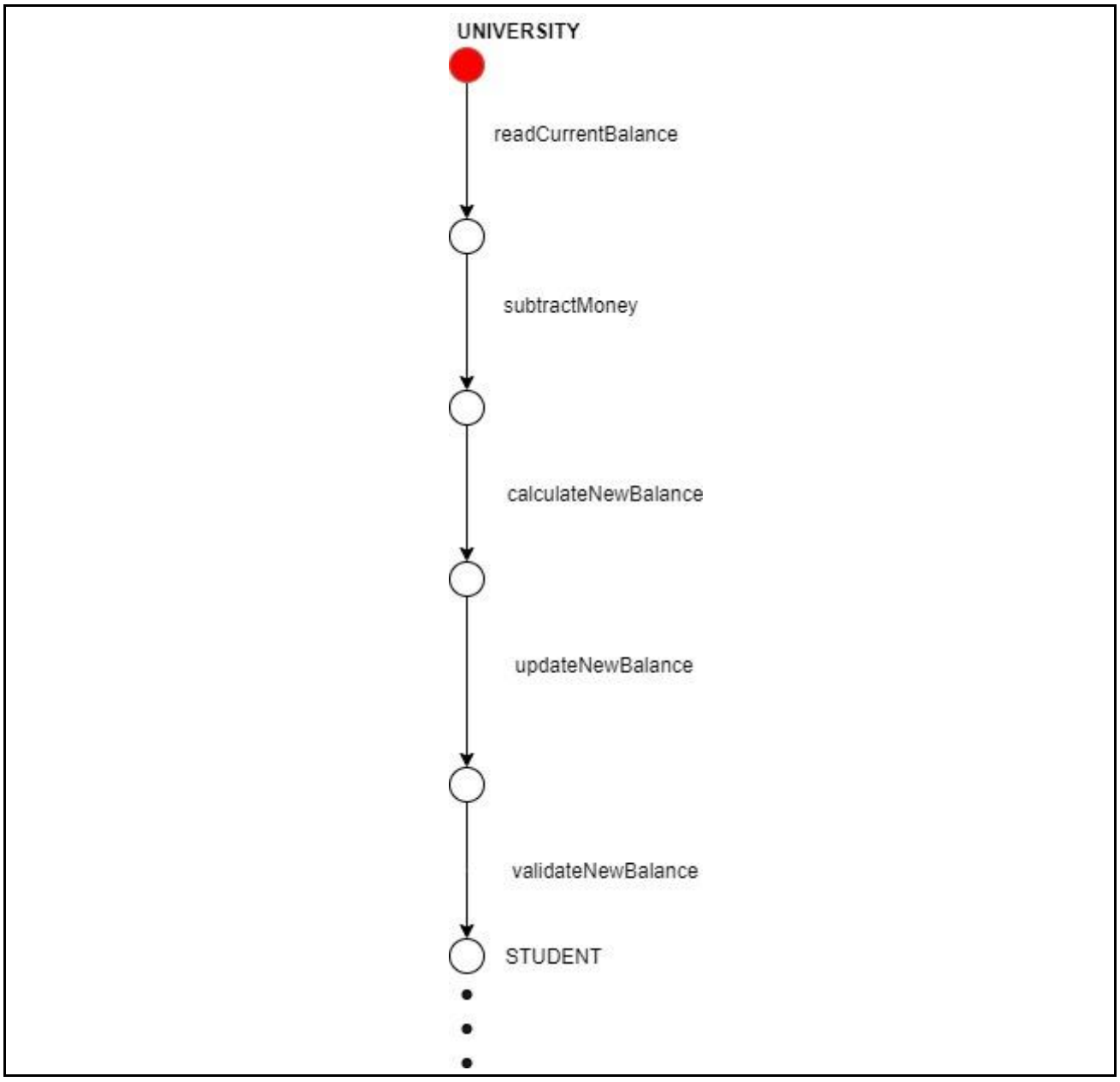
## 5. LTS States

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.)

| States | Represents |
|---|---|
| 0 | University reads the bank account balance |
| 1 | withdrawal money amount from the bank account balance is subtracted from the balance |
| 2 | The bank account balance after the withdrawal of money is calculated |
| 3 | Bank account balance after the withdrawal of money is updated |
| 4 | Updated bank account balance after the withdrawal is validated. |
| | |
| | |
| | |
| | |
| | |
| | |

## 6. Trace Tree for FSP Process

The trace tree for the process. Use the conventions given in the lecture notes.



## 1. FSP Process Attributes

| Attribute | Value |
|---|---|
| **Name** | LOAN_COMPANY |
| **Description** | Money deposit process to the bank account by a loan company |
| **Alphabet** | { readCurrentBalance, addCurrentBalance,calculateNewBalance, updateNewBalance, validateNewBalance} |
| **Number of States** | |
| **Deadlocks (yes/no)** | No |
| **Deadlock Trace(s)** | No |

## 2. FSP Process Code

| **FSP Process:** |
|---|
| LOAN_COMPANY = (readCurrentBalance -> DEPOSIT),<br>DEPOSIT= (addCurrentBalance -> CALCULATE_BALANCE),<br>CALCULATE_BALANCE= (calculateNewBalance-> updateNewBalance ->  validateNewBalance -> LOAN_COMPANY) . |

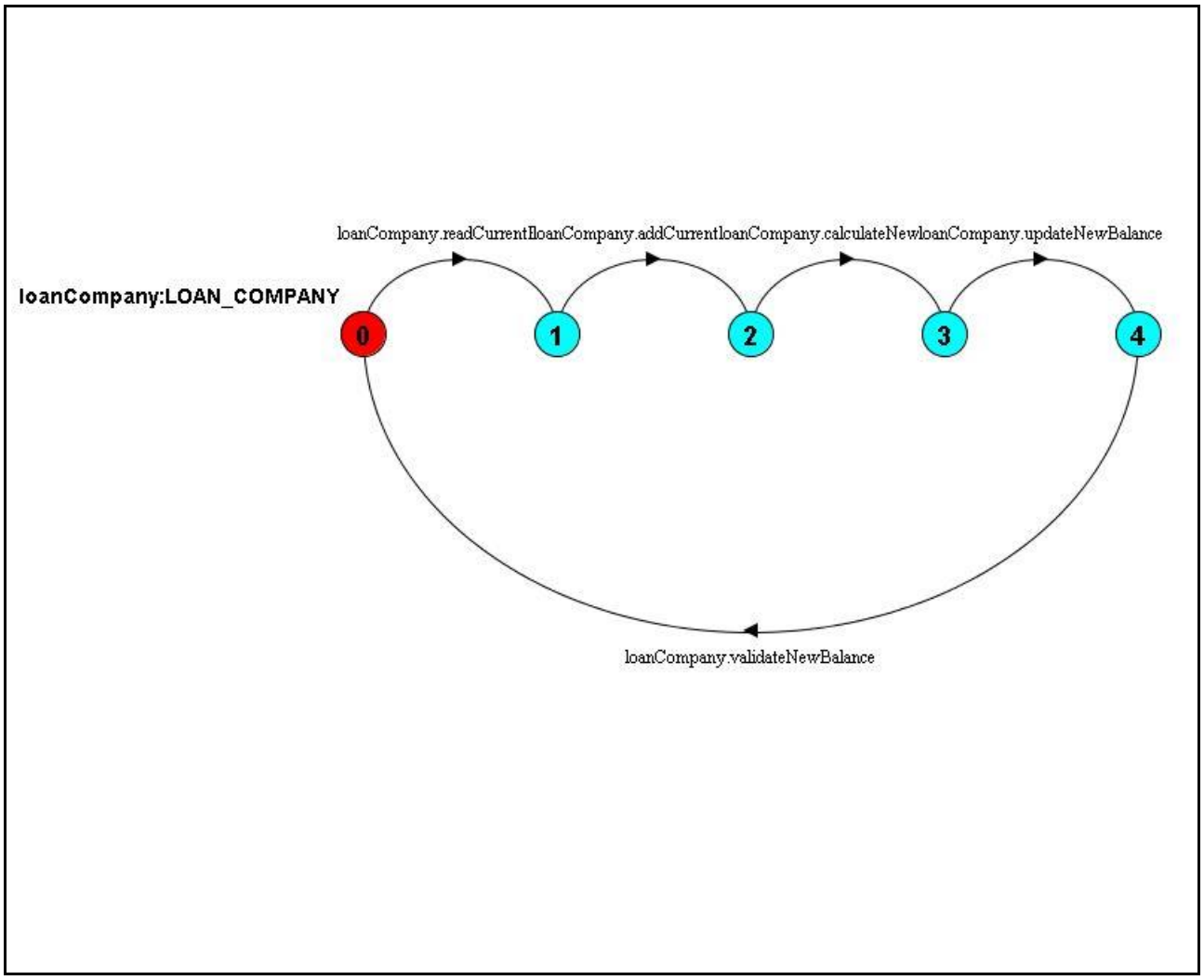
## 3. Actions Description

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 |
|---|---|---|
| readCurrentBalance | Student reads the current bank account balance | Synchronized |
| addCurrentBalance | Add the deposit money amount to the bank account balance | Asynchronous |
| calculateNewBalance | Calculate the bank account balance after depositing the money | Asynchronized |
| updateNewBalance | Update the bank account balance after depositing the money | Synchronized |
| validateNewBalance | Validate the updated bank account balance | Synchronized |
| | | |
| | | |
| | | |

## 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.



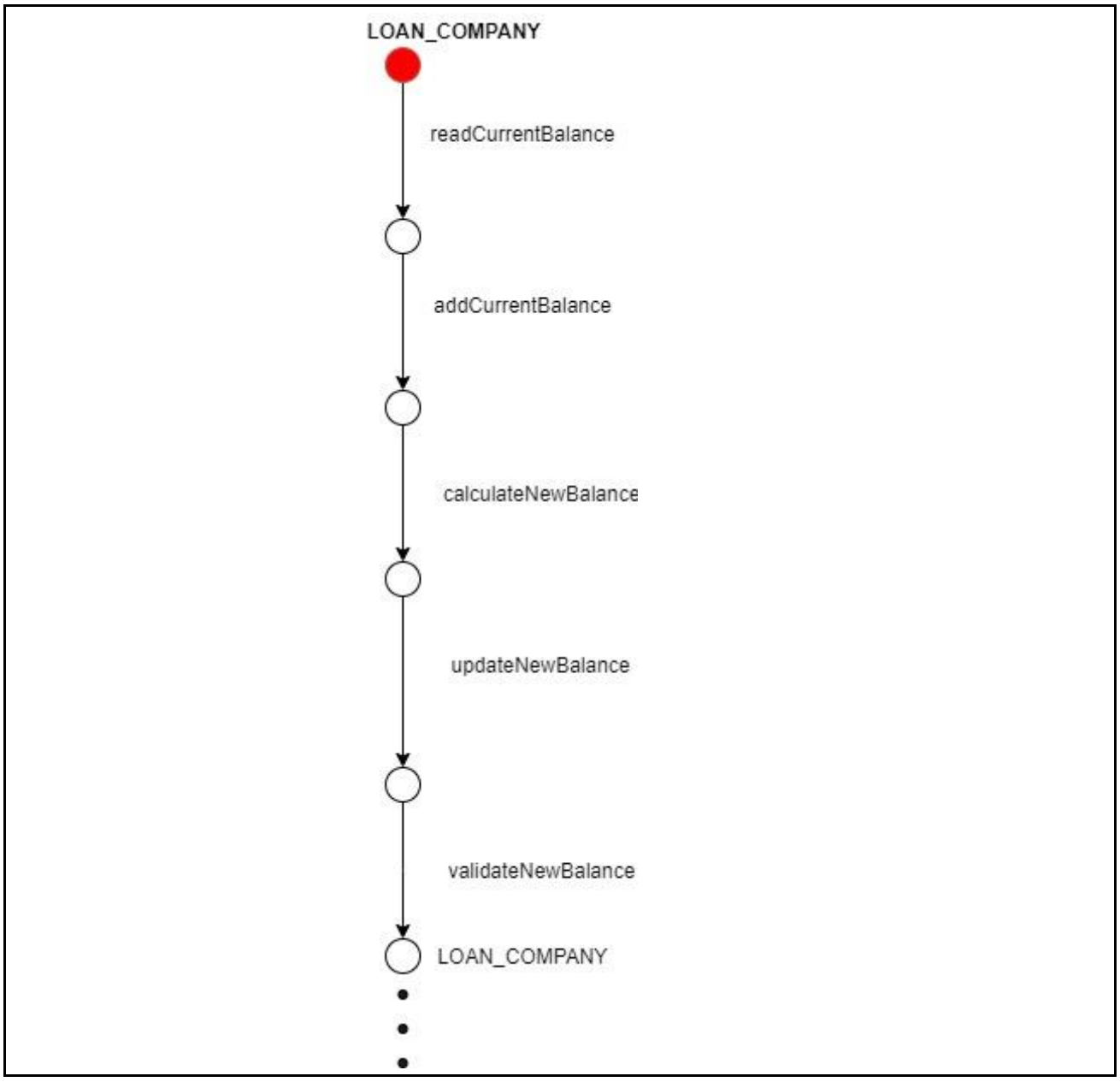
## 5. LTS States

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.)

| States | Represents |
|---|---|
| 0 | Loan company reads student's bank account balance |
| 1 | Deposit money amount to the bank account is added to the bank account balance |
| 2 | The bank account balance after depositing the money is calculated |
| 3 | Bank account balance after depositing the money is updated |
| 4 | Updated bank account balance after the money deposit is validated. |
| | |
| | |
| | |
| | |
| | |
| | |

## 6. Trace Tree for FSP Process

The trace tree for the process. Use the conventions given in the lecture notes.



# FSP Process Composition Analysis & Design Form

## 1. FSP Composition Process Attributes

| Attribute | Value |
|---|---|
| **Name** | BANKING_SYSTEM |
| **Description** | Process of a banking system which allow multiple parties to interact with student's bank account |
| **Alphabet** | { loanCompany.{ addCurrentBalance, calculateCurrentBalance, readCurrentBalance, {updateCurrentBalance, validateCurrentBalance}}, granny.{addCurrentBalance, calculateCurrentBalance, readCurrentBalance, sendCard, {updateCurrentBalance, validateCurrentBalance}}, student.{buyPhone, calculateCurrentBalance, readCurrentBalance, subtractMoney, {updateCurrentBalance, validateCurrentBalance}}, university.{calculateCurrentBalance,readCurrentBalance, subtractMoney, {updateCurrentBalance, validateCurrentBalance}} } |
| **Sub-processes** | STUDENT,GRANDMOTHER,UNIVERSITY,LOAN_COMPANY |
| **Number of States** | 19 |
| **Number of Transitions** | 16 |
| **Deadlocks (yes/no)** | No |
| **Deadlock Traces** | No |

## 2. FSP "main" Program Code

**The code for the parallel composition of all of the sub-processes and the definitions of any process labelling sets used. (Do not include the code for the sub-processes.)**

| **FSP Program:** |
|---|

```
||    BANKING_SYSTEM    =     (student    :     STUDENT
                                ||    grandmother :     GRANDMOTHER

                                ||    loanCompany :     LOAN_COMPANY

                                ||    university  :     UNIVERSITY

                                ||    {student,    grandmother,
      loanCompany,      university} ::    BANK_ACCOUNT)     .
```

## 3. Combined Sub-processes

| Process | Description |
|---|---|
| STUDENT | Money withdrawal process from a bank account by a student and buy a smart phone. |
| GRANDMOTHER | Money deposit process to the student's bank account by a his/her grandmother as a birthday present and send e birthday card. |
| LOANCOMPANY | Money deposit process to the bank account by a loan company |
| UNIVERSITY | Money withdrawal process from the bank account by the University |
| | |
| | |

## 4. Analysis of Combined Process Actions

**Asynchronous** actions are done independently by a single sub-process & the **synchronous** actions are performed by at least two sub-process in the combination.

**Blocked** actions are ones that could be performed by at least one sub-process, but because it has been added to the alphabet of another that can not perform it, the action becomes synchronised, but

it can not be performed.

**Hidden** actions are those that are internal to the combined process & do not form part of the visible interface.

| Synchronous Actions | Synchronised by Sub-Processes |
|---|---|
| student. readCurrentBalance, student.calculateCurrentBalance, student.updateCurrentBalance, student .validateCurrentBalance | STUDENT,BANK_ACCOUNT |
| granny. readCurrentBalance, granny.calculateCurrentBalance, granny.updateCurrentBalance, granny.validateCurrentBalance | GRANDMOTHER,BANK_ACCOUNT |
| loanCompany. readCurrentBalance, loanCompany.calculateCurrentBalance, loanCompany.updateCurrentBalance, loanCompany.validateCurrentBalance | LOANCOMPANY,BANK_ACCOUNT |
| | |
| | |
| | |

| Asynchronous Actions | Performed by Sub-Process |
|---|---|
| buyPhone | STUDENT |
| subtractMoney | STUDENT,UNIVERSITY |
| addCurrentBalance | GRANDMOTHER,LOAN-COMPANY |
| sendCard | GRANDMOTHER |
| | |
| | |

## 5. Composition Structure Diagram

The structure diagram for the parallel composition.