INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

in collaboration with

the University of Westminster, UK

BEng. (Hons) in Software Engineering

6SENG002W

FSP Process Design Form

Name	Ekanayaka **Devon** Christian Nileesha Wijesinghe
IIT Student ID	2016319
UoW Student ID	w1654187
Date	02/01/2020

Table of Content

1. Student Process **3**
1.1. FSP Process Attributes 3
1.2. FSP Process Code 3
1.3. Actions Description 4
1.4. FSM/LTS Diagrams of FSP Process 5
1.5. LTS States 5
1.6. Trace Tree for FSP Process 6

2. Grand Mother Process **7**
2.1. FSP Process Attributes 7
2.2. FSP Process Code 7
2.3. Actions Description 8
2.4. FSM/LTS Diagrams of FSP Process 9
2.5. LTS States 9
2.6. Trace Tree for FSP Process 10

3. Loan Company Process **11**
3.1. FSP Process Attributes 11
3.2. FSP Process Code 11
3.3. Actions Description 12
3.4. FSM/LTS Diagrams of FSP Process 13
3.5. LTS States 13
3.6. Trace Tree for FSP Process 14

4. University Process **15**
4.1. FSP Process Attributes 15
4.2. FSP Process Code 15
4.3. Actions Description 16
4.4. FSM/LTS Diagrams of FSP Process 17
4.5. LTS States 17
4.6. Trace Tree for FSP Process 18

5. Bank Account Process **19**
5.1. FSP Process Attributes 19
5.2. FSP Process Code 19
5.3. Actions Description 20
5.4. FSM/LTS Diagrams of FSP Process 21

5.5. LTS States 21
5.6. Trace Tree for FSP Process 22

6. Banking System Process 23
6.1. FSP Composition Process Attributes 23
6.2. FSP "main" Program Code 24
6.3. Combined Sub-processes 24
6.4. Analysis of Combined Process Actions 25
6.5. FSM/LTS Diagrams of FSP Process 27
6.6. Composition Structure Diagram 28

1. Student Process

1.1. FSP Process Attributes

Attribute	Value
Name	STUDENT
Description	This process simulates the actions taken by the student to interact with the bank account and buy phone
Alphabet	**alphabet(STUDENT)** = { bankAcc.{ calculateNewBalance[-1..3], insufficientBalance, readBalance[1], updateBalance[-1..3], withdraw[1..2] }, buySamsungPhone }
Number of States	6
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

1.2. FSP Process Code

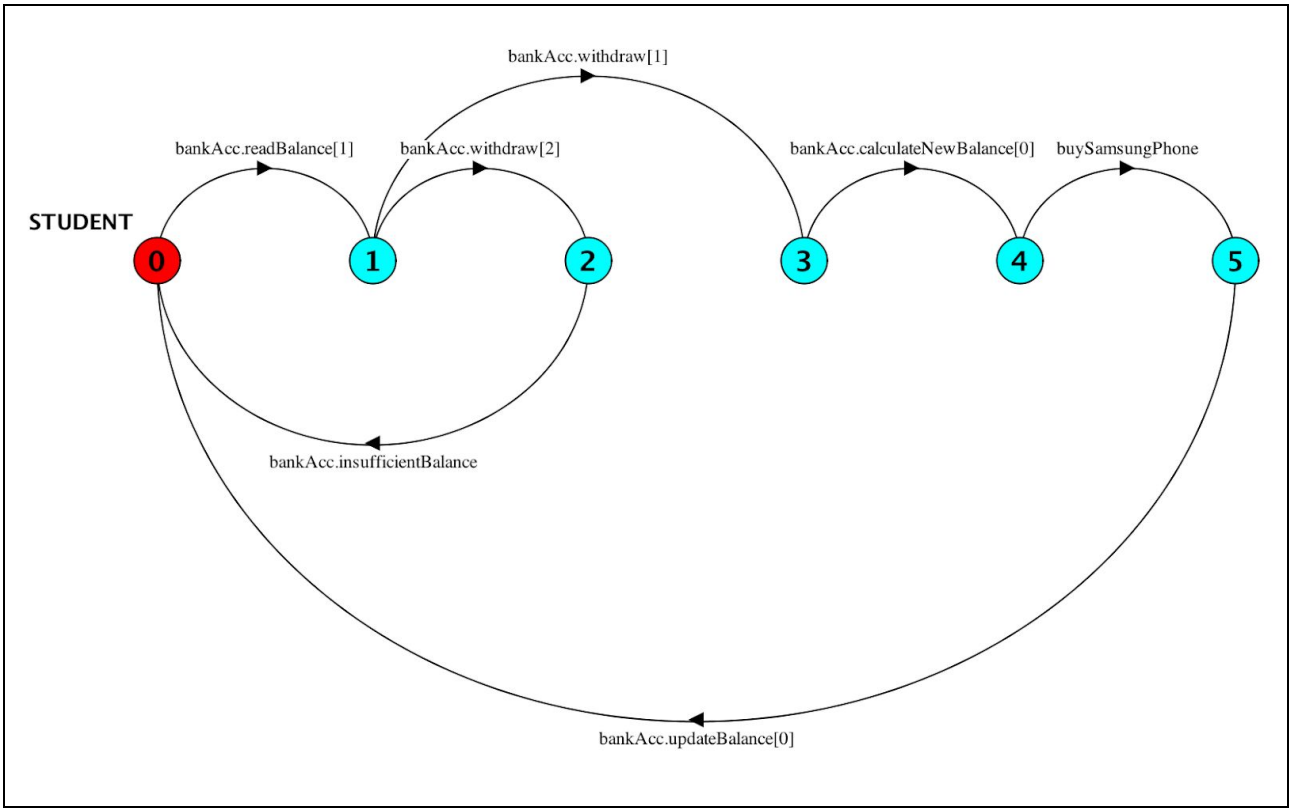
FSP Process:

```
/* STUDENT PROCESS */
STUDENT = INITIAL,
INITIAL = (bankAcc.readBalance[initBal: INITIAL_BALANCE] -> PERFORM_WITHDRAWAL[initBal]),
PERFORM_WITHDRAWAL[initBal: INITIAL_BALANCE] = ( bankAcc.withdraw[amount: TRANSACTION_RANGE] ->
CALCULATE_NEW_BALANCE[initBal][amount] ),
CALCULATE_NEW_BALANCE[initBal: INITIAL_BALANCE][amount: TRANSACTION_RANGE] = (
   when (amount > initBal) bankAcc.insufficientBalance -> STUDENT |
   when (amount <= initBal) bankAcc.calculateNewBalance[initBal - amount] -> BUY_PHONE[initBal -
amount]
),
BUY_PHONE[finalbal: FINAL_BALANCE] = ( buySamsungPhone -> UPDATE_BALANCE[finalbal] ),
UPDATE_BALANCE[finalbal: FINAL_BALANCE] = ( bankAcc.updateBalance[finalbal] -> STUDENT ) +
BankAccountExtention .
```

1.3. Actions Description

Actions	Represents	Synchronous or Asynchronous
bankAcc.readBalance	A process at **INITIAL** state can invoke the *bankAcc.readBalance* action to transition to **PERFORM_WITHDRAWAL** state	Synchronous
bankAcc.withdraw	A process at **PERFORM_WITHDRAWAL** state can invoke the *bankAcc.withdraw* action to transition to **CALCULATE_NEW_BALANCE** state	Synchronous
bankAcc.insufficientBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.insufficientBalance* action to transition to **INITIAL** state	Synchronous
bankAcc.calculateNewBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.calculateNewBalance* action to transition to **BUY_PHONE** state	Synchronous
buySamsungPhone	A process at **BUY_PHONE** state can invoke the *buySamsungPhone* action to transition to **UPDATE_BALANCE** state	Asynchronous
bankAcc.updateBalance	A process at **UPDATE_BALANCE** state can invoke the *bankAcc.updateBalance* action to transition to **INITIAL** state	Synchronous

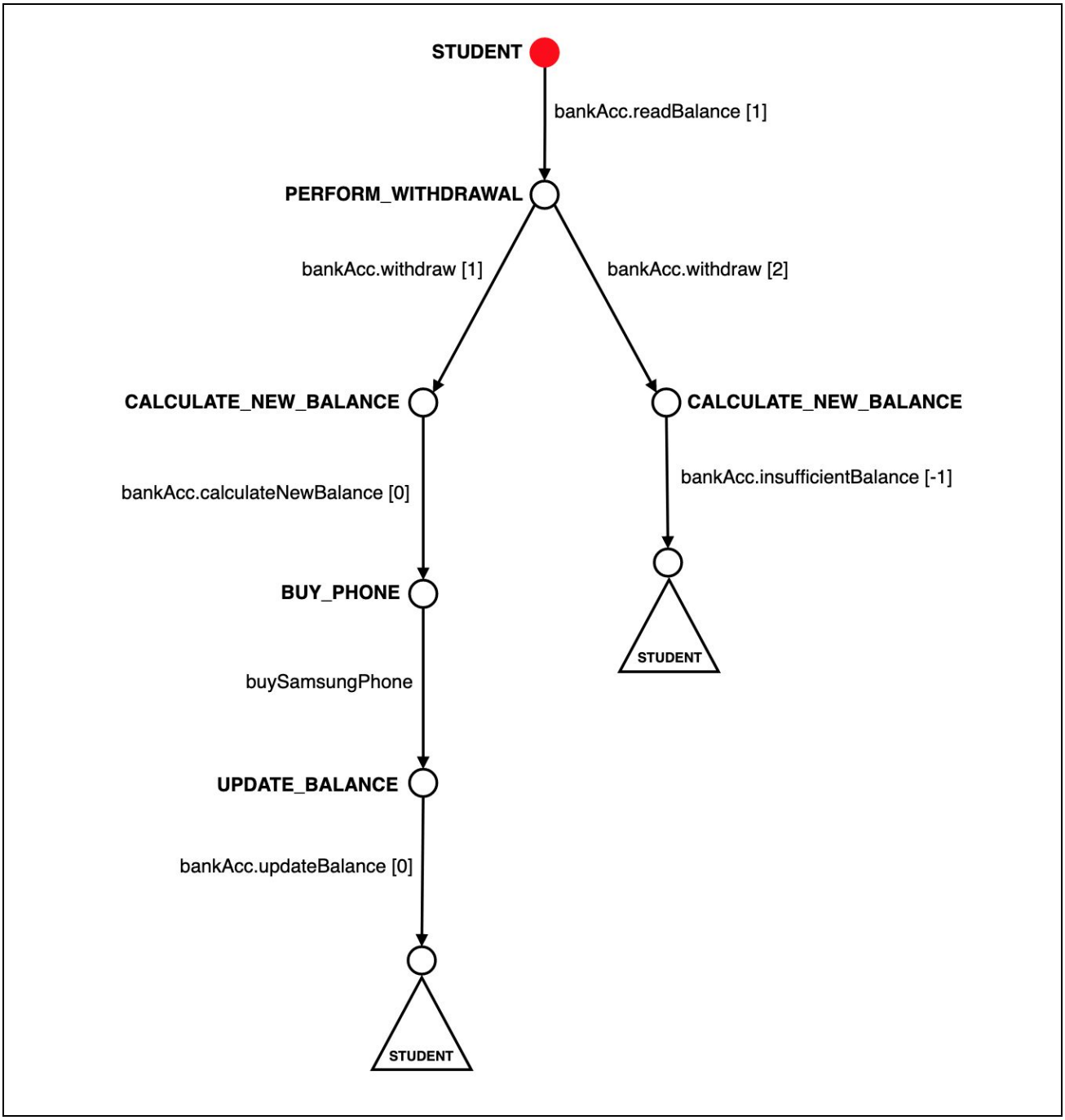
1.4. FSM/LTS Diagrams of FSP Process



1.5. LTS States

States	Represents
INITIAL	Initially starting state and also can be be state after invoking *bankAcc.updateBalance* or *bankAcc.insufficientBalance* action
PERFORM_WITHDRAWAL	State after invoking *bankAcc.readBalance* action
CALCULATE_NEW_BALANCE	State after invoking *bankAcc.withdraw* action
BUY_PHONE	State after invoking *bankAcc.calculateNewBalance* action
UPDATE_BALANCE	State after invoking *buySamsungPhone* action

1.6. Trace Tree for FSP Process



2. Grand Mother Process

2.1. FSP Process Attributes

Attribute	Value
Name	GRAND_MOTHER
Description	This process simulates the actions taken by the grandmother to interact with the bank account to deposit birthday present money and then send send E-Birthday Card
Alphabet	**alphabet(GRAND_MOTHER)** = { bankAcc.{calculateNewBalance [-1..3], depositBirthdayPresentMoney[1..2], readBalance[1], transactionIsInvalid, updateBalance[-1..3] }, sendEBirthdayCard }
Number of States	7
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

2.2. FSP Process Code

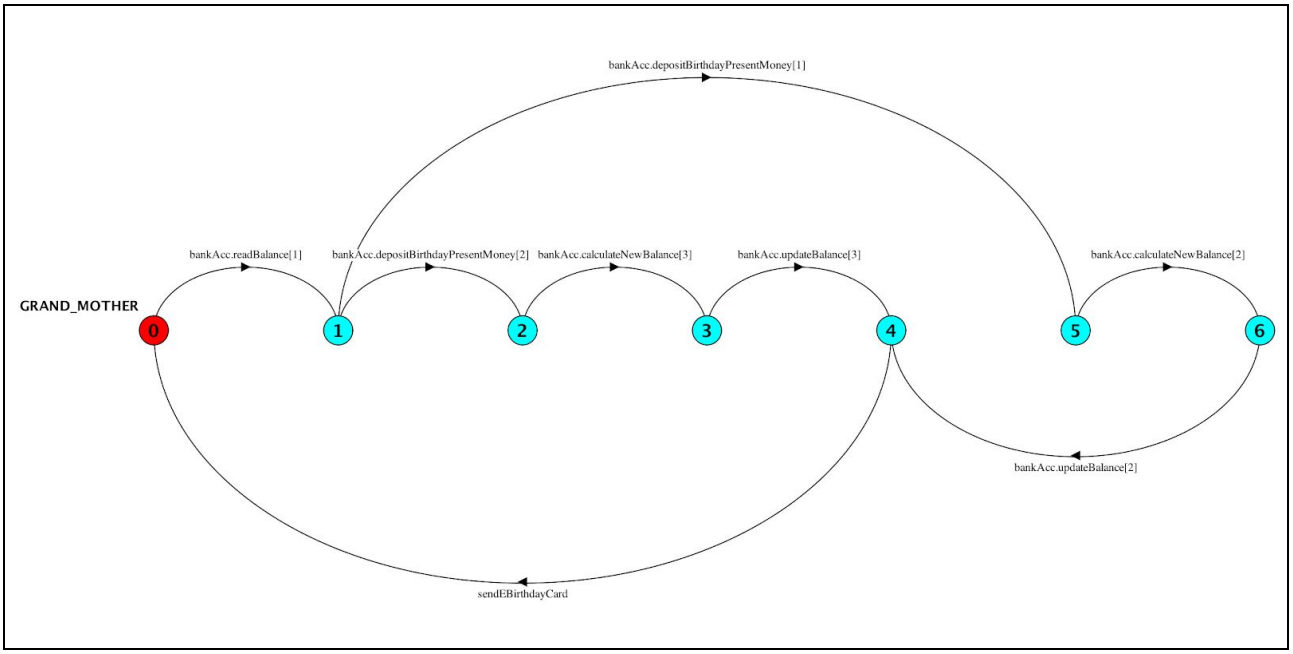
FSP Process:

```
/* GRAND MOTHER PROCESS */
GRAND_MOTHER = INITIAL,
INITIAL = ( bankAcc.readBalance[initBal: INITIAL_BALANCE] -> PERFORM_DEPOSIT[initBal] ),
PERFORM_DEPOSIT[initBal: INITIAL_BALANCE] = ( bankAcc.depositBirthdayPresentMoney[amount:
TRANSACTION_RANGE] -> CALCULATE_NEW_BALANCE[initBal][amount] ),
CALCULATE_NEW_BALANCE[initBal: INITIAL_BALANCE][amount: TRANSACTION_RANGE] = (
bankAcc.calculateNewBalance[initBal+amount] -> UPDATE_BALANCE[initBal + amount] ),
UPDATE_BALANCE[finalbal: FINAL_BALANCE] = (bankAcc.updateBalance[finalbal] -> SEND_CARD),
SEND_CARD = ( sendEBirthdayCard -> GRAND_MOTHER ) + BankAccountExtention .
```

2.3. Actions Description

Actions	Represents	Synchronous or Asynchronous
bankAcc.readBalance	A process at **INITIAL** state can invoke the *bankAcc.readBalance* action to transition to **PERFORM_DEPOSIT** state	Synchronous
bankAcc.depositBirthdayPrese-ntMoney	A process at **PERFORM_DEPOSIT** state can invoke the *bankAcc.depositBirthdayPres-entMoney* action to transition to **CALCULATE_NEW_BALANCE** state	Synchronous
bankAcc.calculateNewBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.calculateNewBalanc-e* action to transition to **UPDATE_BALANCE** state	Synchronous
bankAcc.updateBalance	A process at **UPDATE_BALANCE** state can invoke the *bankAcc.updateBalance* action to transition to **SEND_CARD** state	Synchronous
sendEBirthdayCard	A process at **SEND_CARD** state can invoke the *sendEBirthdayCard* action to transition to **INITIAL** state	Asynchronous

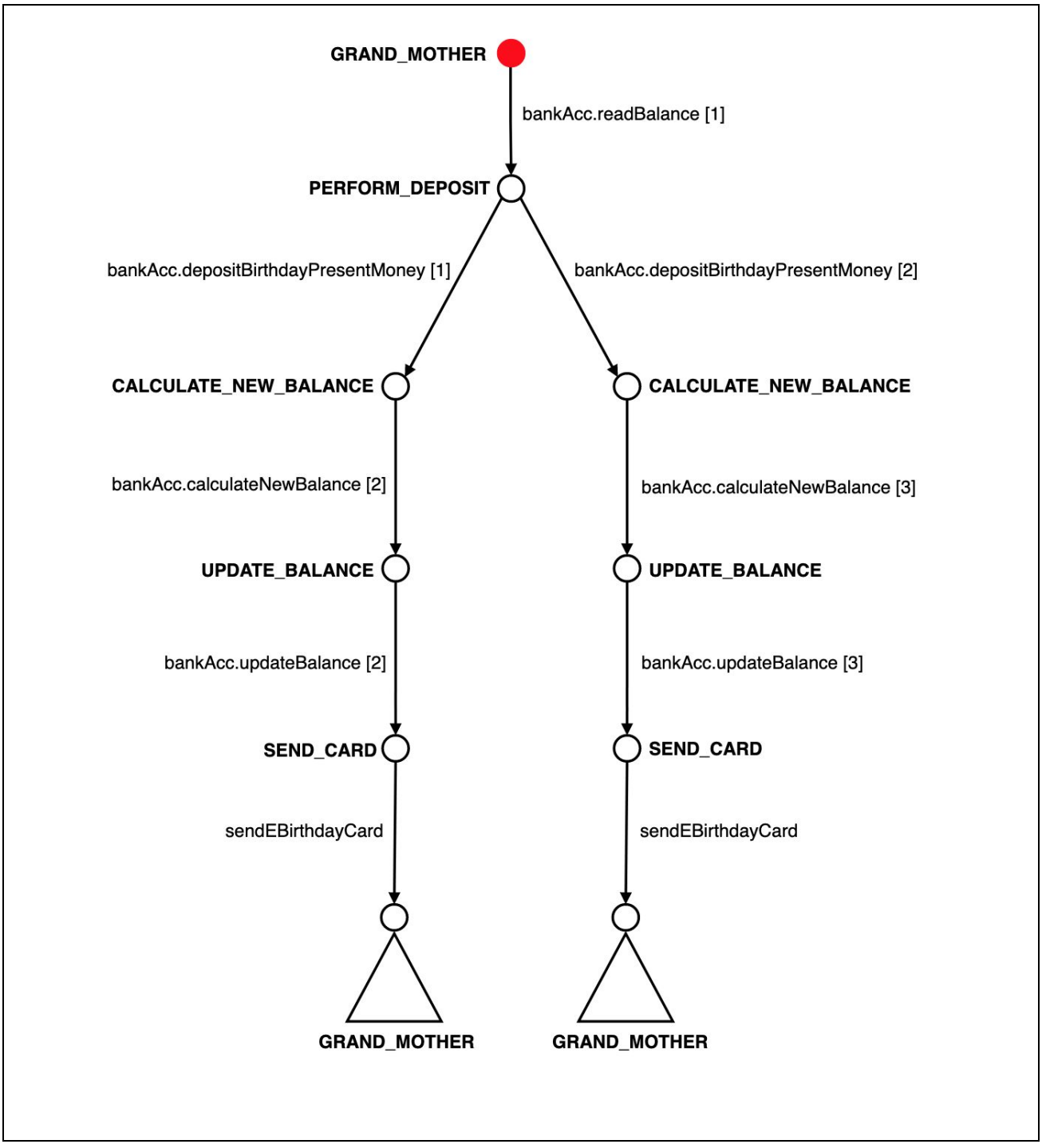
2.4. FSM/LTS Diagrams of FSP Process



2.5. LTS States

States	Represents
INITIAL	Initially starting state and also can be be state after invoking *sendEBirthdayCard* action
PERFORM_DEPOSIT	State after invoking *bankAcc.readBalance* action
CALCULATE_NEW_BALANCE	State after invoking *bankAcc.depositBirthdayPresentMoney* action
UPDATE_BALANCE	State after invoking *bankAcc.calculateNewBalance* action
SEND_CARD	State after invoking *bankAcc.updateBalance* action

2.6. Trace Tree for FSP Process



3. Loan Company Process

3.1. FSP Process Attributes

Attribute	Value
Name	LOAN_COMPANY
Description	This process simulates the actions taken by the loan company to interact with the bank account and deposit loan amount
Alphabet	**alphabet(LOAN_COMPANY)** = { bankAcc.{ calculateNewBalance [-1..3], depositLoan [1..2], readBalance [1], transactionIsInvalid, updateBalance [-1..3] } }
Number of States	6
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

3.2. FSP Process Code

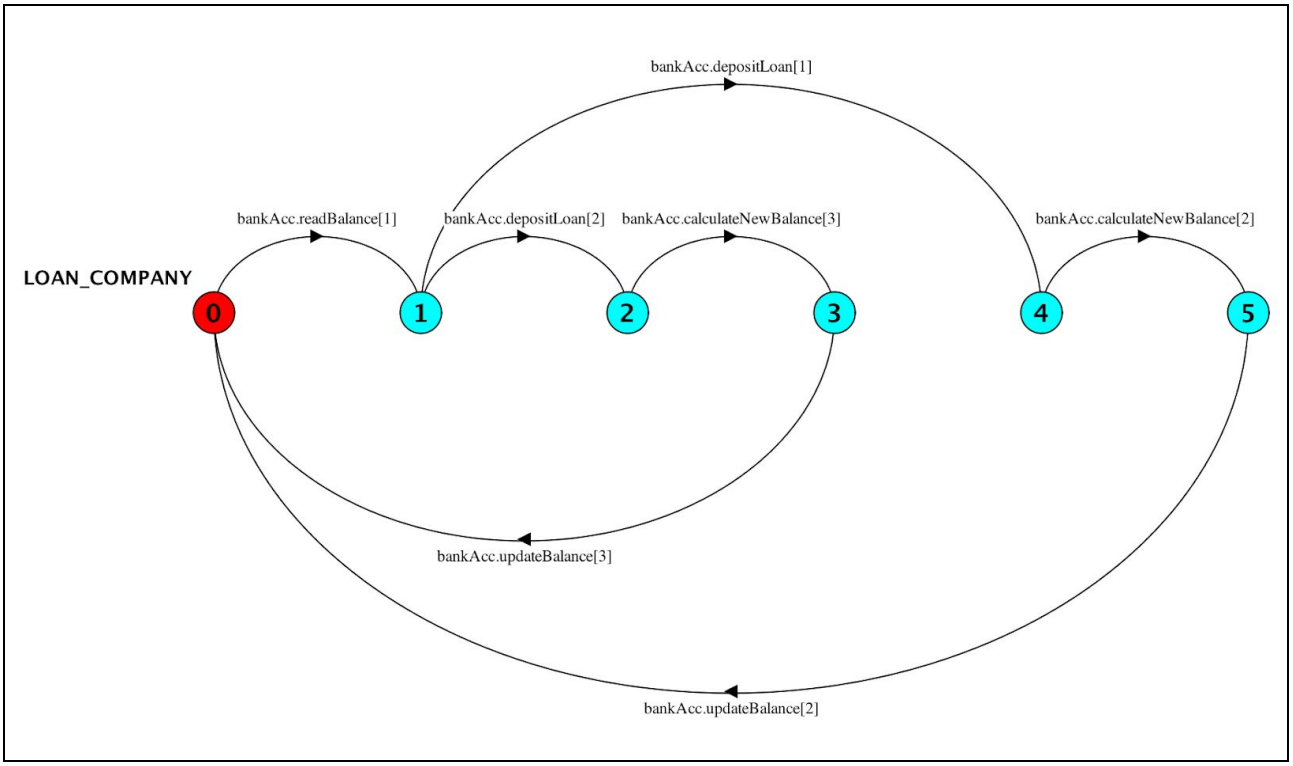
FSP Process:

```
/* LOAN COMPANY PROCESS */
LOAN_COMPANY = INITIAL,
INITIAL = ( bankAcc.readBalance[initBal: INITIAL_BALANCE] -> PERFORM_DEPOSIT[initBal] ),
PERFORM_DEPOSIT[initBal: INITIAL_BALANCE] = ( bankAcc.depositLoan[amount:
TRANSACTION_RANGE] -> CALCULATE_NEW_BALANCE[initBal][amount] ),
CALCULATE_NEW_BALANCE[initBal: INITIAL_BALANCE][amount: TRANSACTION_RANGE] = (
bankAcc.calculateNewBalance[initBal + amount] -> UPDATE_BALANCE[initBal + amount] ),
UPDATE_BALANCE[finalbal: FINAL_BALANCE] = ( bankAcc.updateBalance[finalbal] ->
LOAN_COMPANY ) + BankAccountExtention .
```

3.3. Actions Description

Actions	Represents	Synchronous or Asynchronous
bankAcc.readBalance	A process at **INITIAL** state can invoke the *bankAcc.readBalance* action to transition to **PERFORM_DEPOSIT** state	Synchronous
bankAcc.depositLoan	A process at **PERFORM_DEPOSIT** state can invoke the *bankAcc.depositLoan* action to transition to **CALCULATE_NEW_BALANCE** state	Synchronous
bankAcc.calculateNewBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.calculateNewBalan-ce* action to transition to **UPDATE_BALANCE** state	Synchronous
bankAcc.updateBalance	A process at **UPDATE_BALANCE** state can invoke the *bankAcc.updateBalance* action to transition to **INITIAL** state	Synchronous

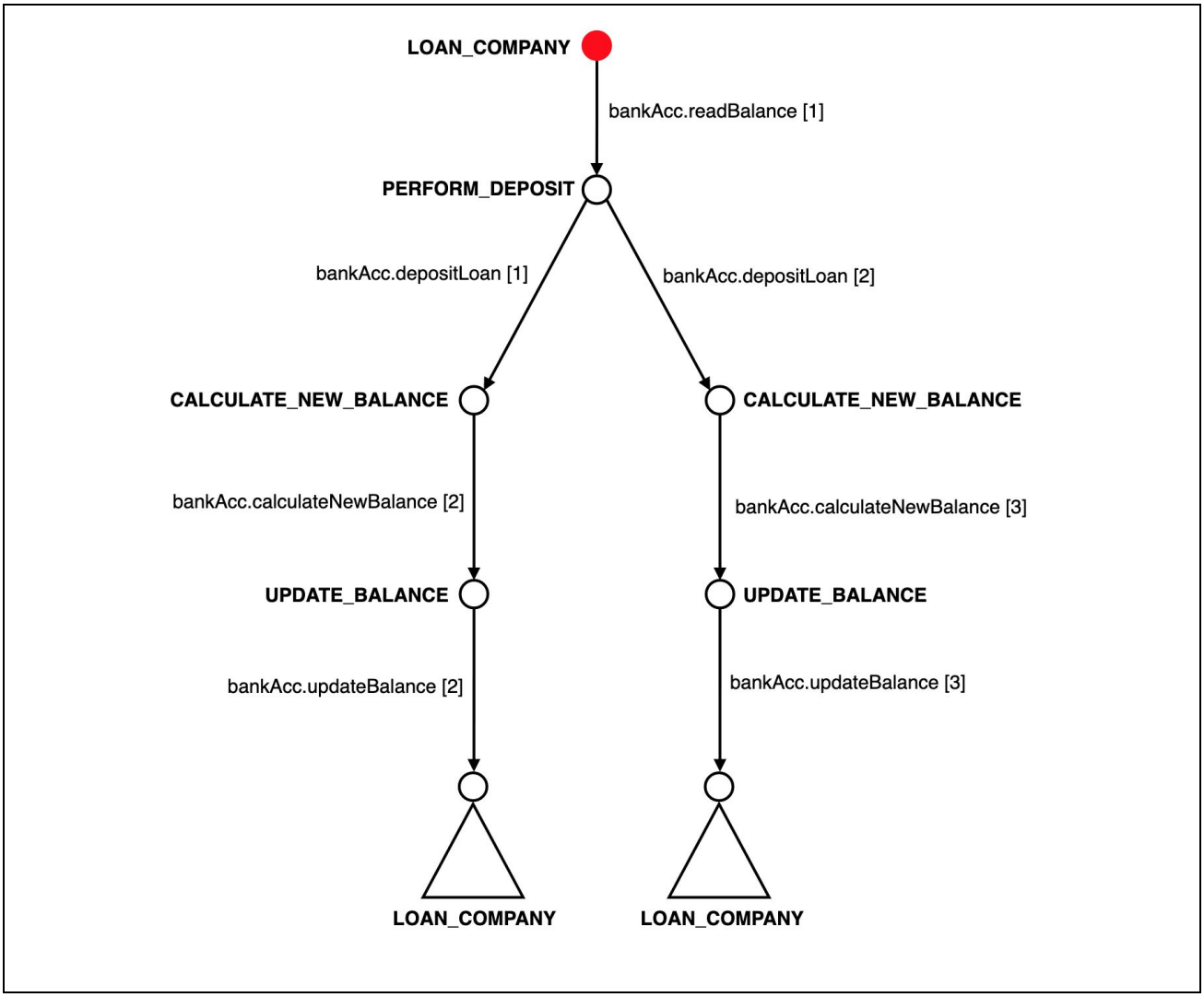
3.4. FSM/LTS Diagrams of FSP Process



3.5. LTS States

States	Represents
INITIAL	Initially starting state and also can be be state after invoking *bankAcc.updateBalance* action
PERFORM_DEPOSIT	State after invoking *bankAcc.readBalance* action
CALCULATE_NEW_BALANCE	State after invoking *bankAcc.depositLoan* action
UPDATE_BALANCE	State after invoking *bankAcc.calculateNewBalance* action

3.6. Trace Tree for FSP Process



4. University Process

4.1. FSP Process Attributes

Attribute	Value
Name	UNIVERSITY
Description	This process simulates the actions taken by the university to interact with the bank account to deduct university fees
Alphabet	**alphabet(UNIVERSITY)** = { bankAcc.{calculateNewBalance [-1..3], deductUniFees [1..2], insufficientBalance, readBalance [1], updateBalance[-1..3] } }
Number of States	5
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

4.2. FSP Process Code

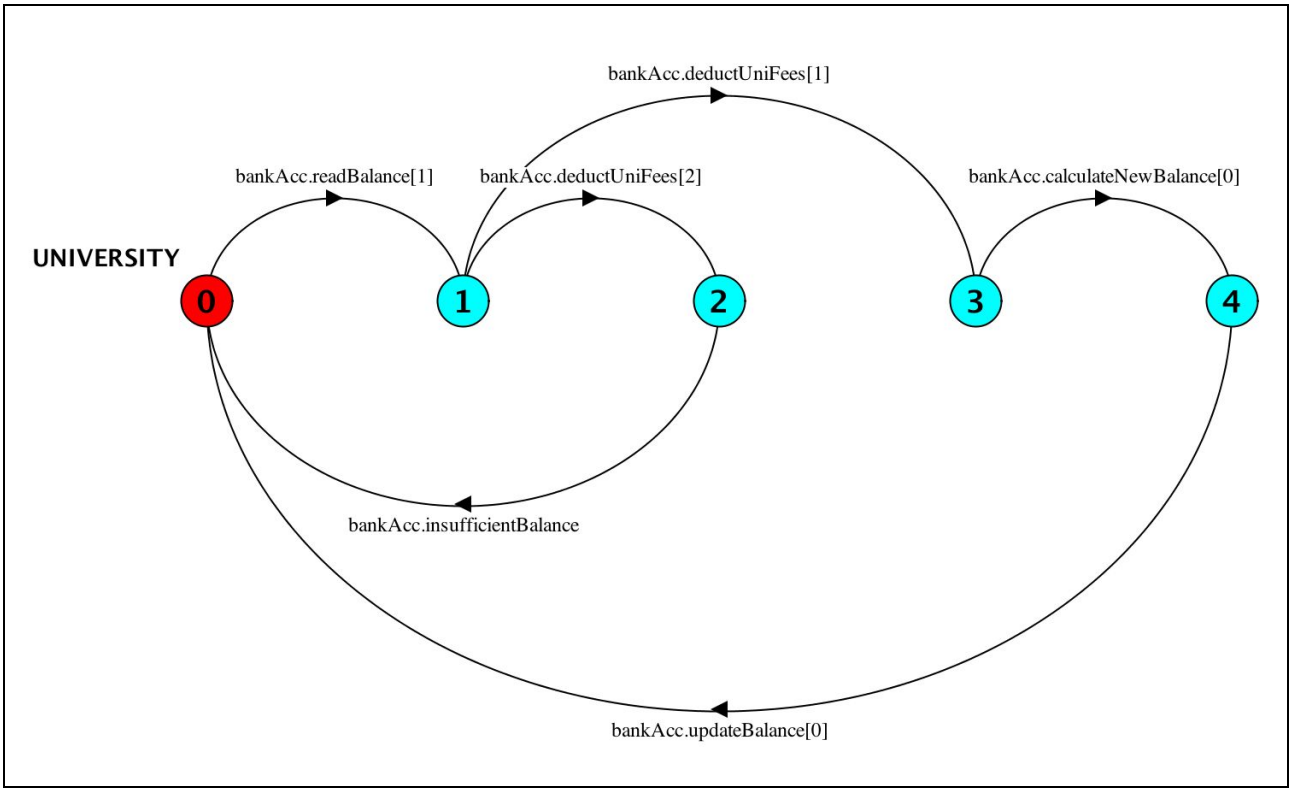
FSP Process:

```
/* UNIVERSITY PROCESS */
UNIVERSITY = INITIAL,
INITIAL = (bankAcc.readBalance[initBal: INITIAL_BALANCE] -> PERFORM_WITHDRAWAL[initBal]),
PERFORM_WITHDRAWAL[initBal: INITIAL_BALANCE] = ( bankAcc.deductUniFees[amount:
TRANSACTION_RANGE] -> CALCULATE_NEW_BALANCE[initBal][amount] ),
CALCULATE_NEW_BALANCE[initBal: INITIAL_BALANCE][amount:TRANSACTION_RANGE] = (
   when (amount > initBal) bankAcc.insufficientBalance -> UNIVERSITY |
   when (amount <= initBal) bankAcc.calculateNewBalance[initBal - amount] ->
UPDATE_BALANCE[initBal - amount]
),
UPDATE_BALANCE[finalbal: FINAL_BALANCE] = ( bankAcc.updateBalance[finalbal] ->
UNIVERSITY) + BankAccountExtention .
```

4.3. Actions Description

Actions	Represents	Synchronous or Asynchronous
bankAcc.readBalance	A process at **INITIAL** state can invoke the *bankAcc.readBalance* action to transition to **PERFORM_WITHDRAWAL** state	Synchronous
bankAcc.deductUniFees	A process at **PERFORM_WITHDRAWAL** state can invoke the *bankAcc.deductUniFees* action to transition to **CALCULATE_NEW_BALANCE** state	Synchronous
bankAcc.insufficientBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.*insufficientBalance action to transition to **INITIAL** state	Synchronous
bankAcc.calculateNewBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *bankAcc.calculateNewBalan-ce* action to transition to **UPDATE_BALANCE** state	Synchronous
bankAcc.updateBalance	A process at **UPDATE_BALANCE** state can invoke the *bankAcc.updateBalance* action to transition to **INITIAL** state	Synchronous

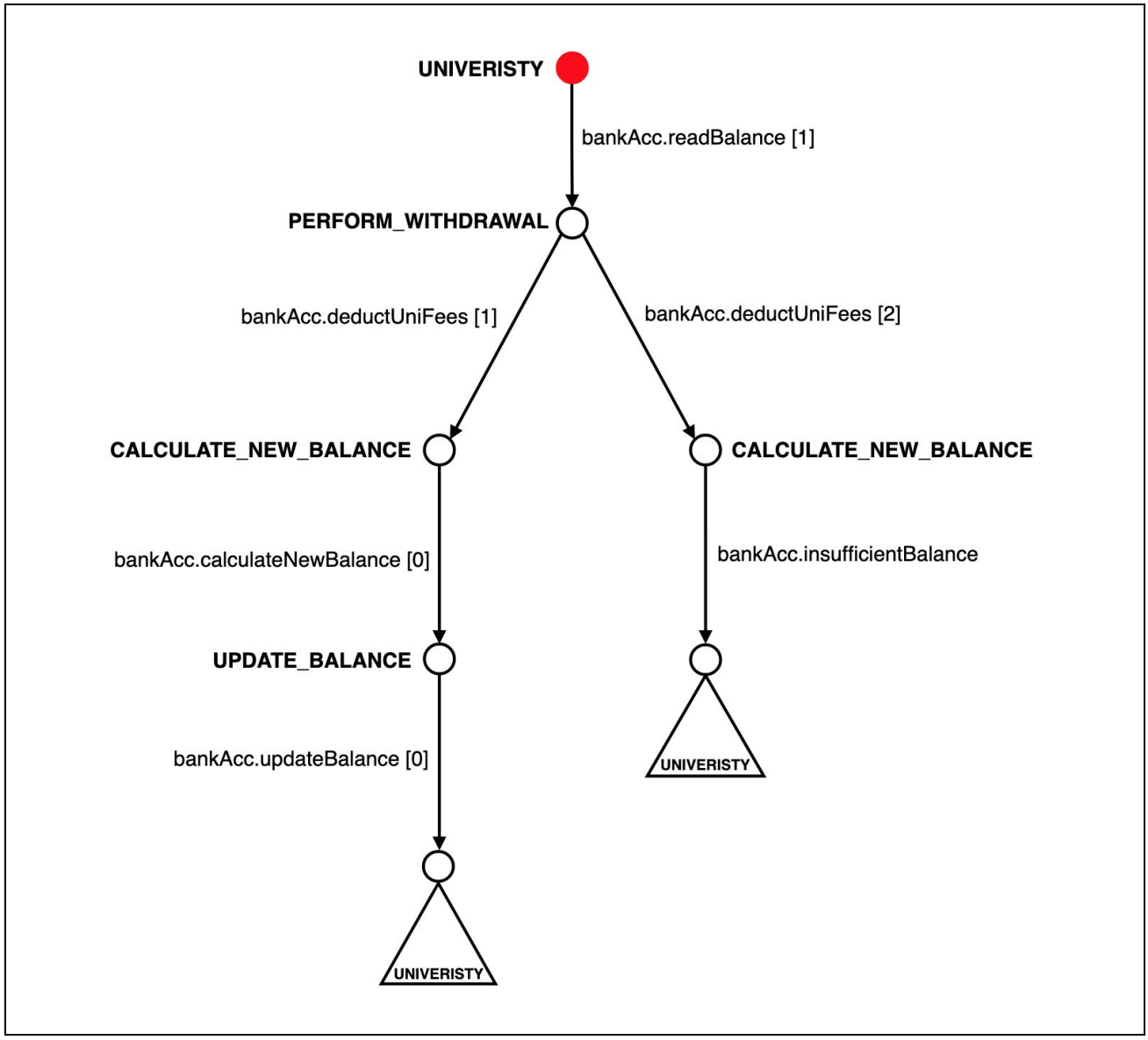
4.4. FSM/LTS Diagrams of FSP Process



4.5. LTS States

States	Represents
INITIAL	Initially starting state and also can be be state after invoking *bankAcc.updateBalance* or *bankAcc.insufficientBalance* action
PERFORM_WITHDRAWAL	State after invoking *bankAcc.readBalance* action
CALCULATE_NEW_BALANCE	State after invoking *bankAcc.deductUniFees* action
UPDATE_BALANCE	State after invoking *bankAcc.calculateNewBalance* action

4.6. Trace Tree for FSP Process



5. Bank Account Process

5.1. FSP Process Attributes

Attribute	Value
Name	BANK_ACCOUNT
Description	This process simulates the actions happening in the bank account. Bank account acts as a shared resource in the banking system.
Alphabet	**alphabet(BANK_ACCOUNT)** = { calculateNewBalance [-1..3], performTransaction [1..2], readBalance [1], transactionIsInvalid, updateBalance [-1..3] }
Number of States	8
Deadlocks (yes/no)	No
Deadlock Trace(s)	N/A

5.2. FSP Process Code

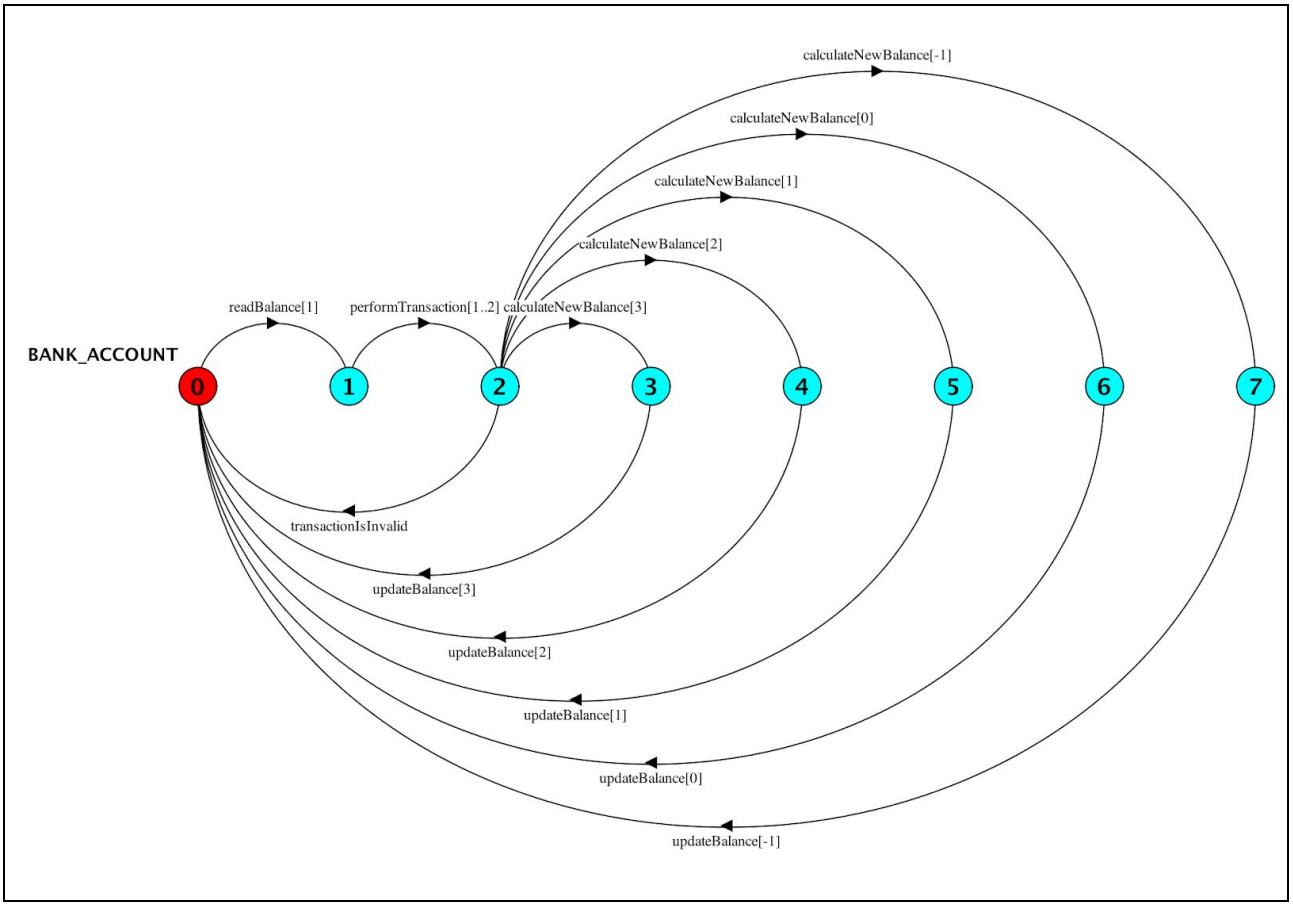
FSP Process:

```
/* BANK ACCOUNT PROCESS */
BANK_ACCOUNT = INITIAL,
INITIAL = ( readBalance[initBal: INITIAL_BALANCE] -> TRANSACTION ),
TRANSACTION = ( performTransaction[TRANSACTION_RANGE] -> CALCULATE_NEW_BALANCE ),
CALCULATE_NEW_BALANCE = ( calculateNewBalance[finalbal: FINAL_BALANCE] ->
UPDATE_BALANCE[finalbal] | transactionIsInvalid -> BANK_ACCOUNT ),
UPDATE_BALANCE[finalbal: FINAL_BALANCE] = ( updateBalance[finalbal] -> BANK_ACCOUNT ) .
```

5.3. Actions Description

Actions	Represents	Synchronous or Asynchronous
readBalance	A process at **INITIAL** state can invoke the *readBalance* action to transition to **TRANSACTION** state	Synchronous
performTransaction	A process at **TRANSACTION** state can invoke the *performTransaction* action to transition to **CALCULATE_NEW_BALANCE** state	Synchronous
calculateNewBalance	A process at **CALCULATE_NEW_BALANCE** state can invoke the *calculateNewBalance* action to transition to **UPDATE_BALANCE** state	Synchronous
transactionIsInvalid	A process at **CALCULATE_NEW_BALANCE** state can invoke the *transactionIsInvalid* action to transition to **INITIAL** state	Synchronous
updateBalance	A process at **UPDATE_BALANCE** state can invoke the *updateBalance* action to transition to **INITIAL** state	Synchronous

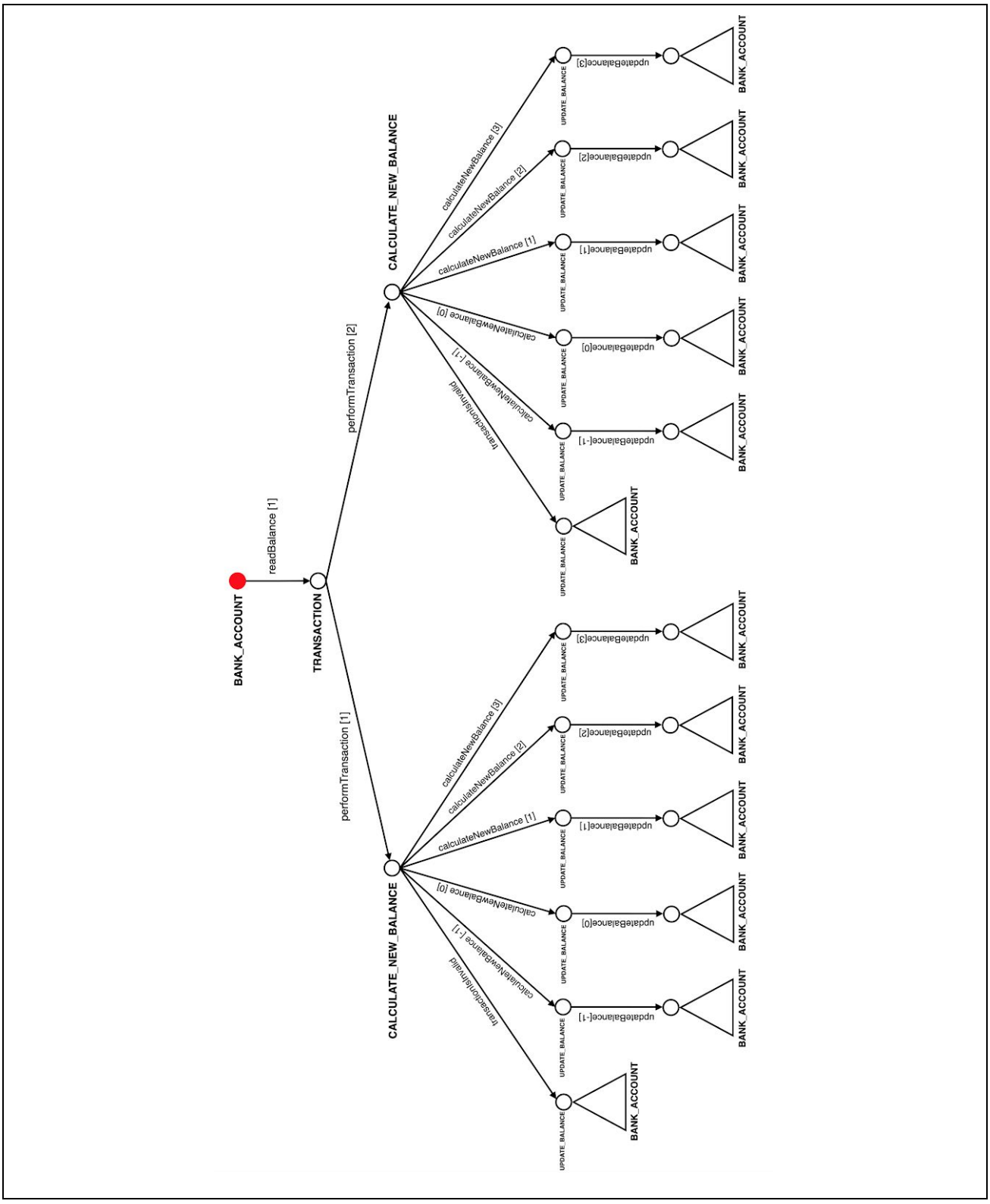
5.4. FSM/LTS Diagrams of FSP Process



5.5. LTS States

States	Represents
INITIAL	Initially starting state and also can be be state after invoking *updateBalance* or *transactionIsInvalid* action
TRANSACTION	State after invoking *readBalance* action
CALCULATE_NEW_BALANCE	State after invoking *performTransaction* action
UPDATE_BALANCE	State after invoking *calculateNewBalance* action

5.6. Trace Tree for FSP Process



6. Banking System Process

6.1. FSP Composition Process Attributes

Attribute	Value
Name	BANKING_SYSTEM
Description	This is the composite process which simulates the complete banking system. The actions done by all the users (student, grandmother, loan company and university) to concurrently interact with the bank account while achieving mutual exclusion is simulated.
Alphabet	**alphabet(BANKING_SYSTEM)** = { grandMother.{ bankAcc.{ calculateNewBalance [-1..3], depositBirthdayPresentMoney [1..2], readBalance [1], transactionIsInvalid, updateBalance [-1..3]}, sendEBirthdayCard }, loanCompany.bankAcc.{ calculateNewBalanc -e [-1..3], depositLoan [1..2], readBalance [1], transactionIsInvalid, updateBalance [-1..3]}, student.{ bankAcc.{ calculateNewBalanc -e[-1..3], insufficientBalance, readBalance [1], updateBalance [-1..3], withdraw[1..2]}, buySamsungPhone}, university.bankAcc.{ calculateNewBalance [-1..3], deductUniFees[1..2], insufficientBalance, readBalance [1], updateBalance[-1..3] }}
Sub-processes	STUDENT, GRAND_MOTHER, LOAN_COMPANY, UNIVERSITY, BANK_ACCOUNT
Number of States	35
Number of Transitions	62
Deadlocks (yes/no)	No
Deadlock Traces	N/A

6.2. FSP "main" Program Code

FSP Program:

```
set AllUsers = { student, grandMother, loanCompany, university }

/* CONCURRENT BANKING SYSTEM PROCESS */
|| BANKING_SYSTEM = (
   student : STUDENT ||
   grandMother : GRAND_MOTHER ||
   loanCompany : LOAN_COMPANY ||
   university : UNIVERSITY ||
   AllUsers :: bankAcc : BANK_ACCOUNT
) / {
   /* Re-labelling actions to achieve synchronization */
   student.bankAcc.withdraw / student.bankAcc.performTransaction,
   student.bankAcc.insufficientBalance / student.bankAcc.transactionIsInvalid,
   grandMother.bankAcc.depositBirthdayPresentMoney /
grandMother.bankAcc.performTransaction,
   loanCompany.bankAcc.depositLoan / loanCompany.bankAcc.performTransaction,
   university.bankAcc.deductUniFees / university.bankAcc.performTransaction,
   university.bankAcc.insufficientBalance / university.bankAcc.transactionIsInvalid
} .
```

6.3. Combined Sub-processes

Process	Description
STUDENT	Process which simulates the actions taken by the student to interact with the bank account and buy phone
GRANDMOTHER	Process which simulates the actions taken by the grandmother to interact with the bank account to deposit birthday present money and then send send E-Birthday Card
LOAN_COMPANY	Process which simulates the actions taken by the loan company to interact with the bank account and deposit loan amount

UNIVERSITY	Process which simulates the actions taken by the university to interact with the bank account to deduct university fees
BANK_ACCOUNT	Process which simulates the actions happening in the bank account. Bank account acts as a shared resource in the banking system.

6.4. Analysis of Combined Process Actions

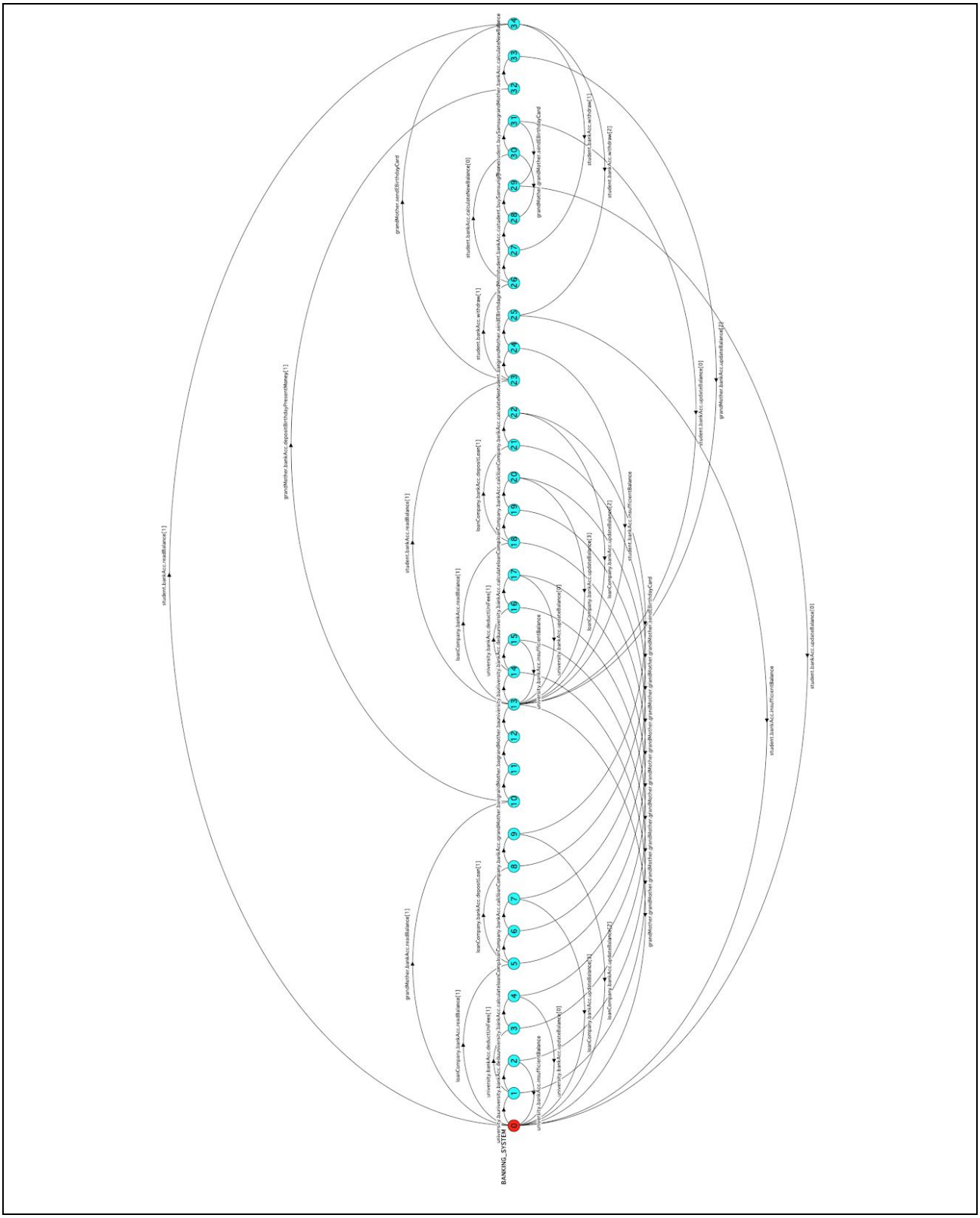
Synchronous Actions	Synchronised by Sub-Processes
student.bankAcc.readBalance[1] student.bankAcc.withdraw[1..2] student.bankAcc.insufficientBalance student.bankAcc.calculateNewBalance[0] student.bankAcc.updateBalance[0]	STUDENT, BANK_ACCOUNT
grandMother.bankAcc.readBalance[1] grandMother.bankAcc.depositBirthdayPresentMoney[1..2] grandMother.bankAcc.calculateNewBalance[2..3] grandMotherbankAcc.updateBalance[2..3]	GRANDMOTHER, BANK_ACCOUNT
loanCompany.bankAcc.readBalance[1] loanCompany.bankAcc.depositLoan[1..2] loanCompany.bankAcc.calculateNewBalance[2..3] loanCompany.bankAcc.updateBalance[2..3]	LOAN_COMPANY, BANK_ACCOUNT
university.bankAcc.readBalance[1] university.bankAcc.deductUniFees[1..2] university.bankAcc.insufficientBalance university.bankAcc.calculateNewBalance[0] university.bankAcc.updateBalance[0]	UNIVERSITY, BANK_ACCOUNT

Asynchronous Actions	Performed by Sub-Process
student.buySamsungPhone	STUDENT
grandMother.sendEBirthdayCard	GRANDMOTHER

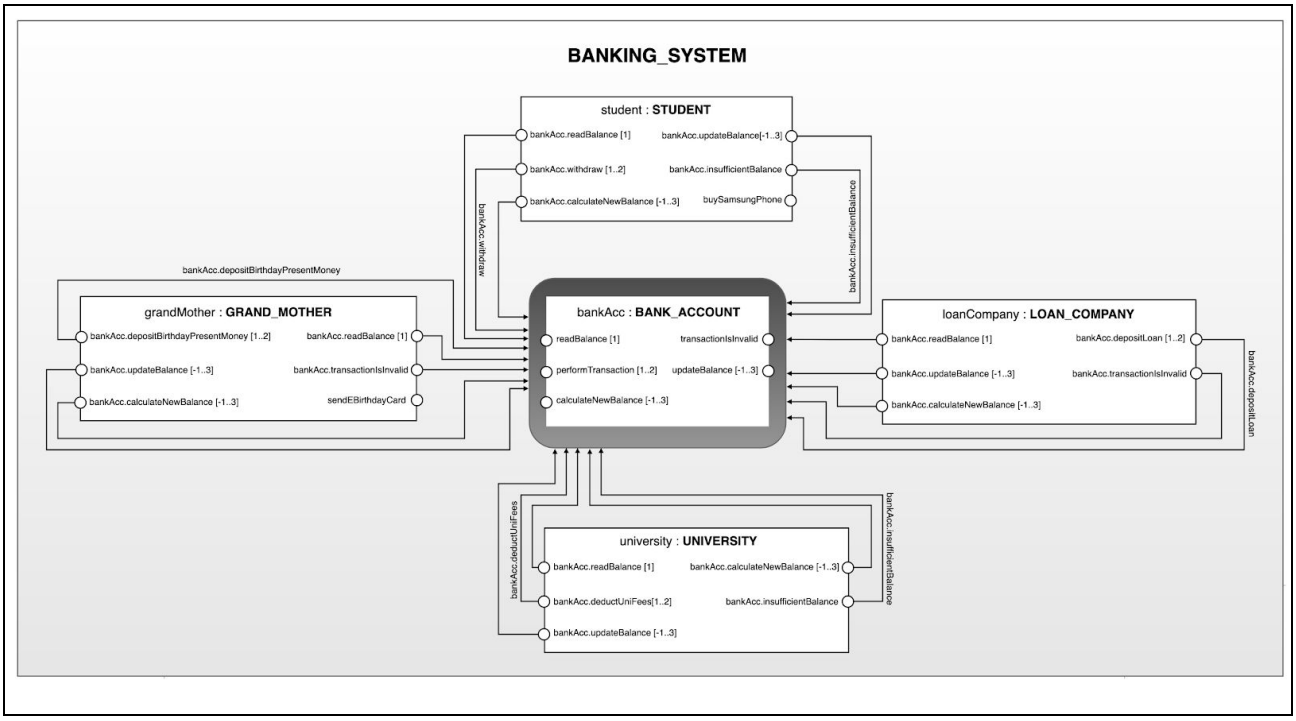
Blocked Action	Blocked by Sub-Processes
student.bankAcc.calculateNewBalance[-1] student.bankAcc.calculateNewBalance[1..3] student.bankAcc.updateBalance[-1] student.bankAcc.updateBalance[1..3]	STUDENT
grandMother.bankAcc.calculateNewBalance[-1..1] grandMother.bankAcc.updateBalance[-1..1] grandMother.bankAcc.transactionIsInvalid	GRANDMOTHER
loanCompany.bankAcc.calculateNewBalance[-1..1] loanCompany.bankAcc.updateBalance[-1..1] loanCompany.bankAcc.transactionIsInvalid	LOAN_COMPANY
university.bankAcc.calculateNewBalance[-1] university.bankAcc.calculateNewBalance[1..3] university.bankAcc.updateBalance[-1] university.bankAcc.updateBalance[1..3]	UNIVERSITY

Internal Action	Performed by Sub-Processes
N/A	N/A

6.5. FSM/LTS Diagrams of FSP Process



6.6. Composition Structure Diagram



------ End of report ------