6SENG002W Concurrent Programming

FSP Process Composition Analysis & Design Form

Name	Kristen Hayzelden
Student ID	W1508466
Date	01/01/2018

1. FSP Composition Process Attributes

Attribute	**Value**
Name	Banking System
Description	Models a Student, Parent, University, and Loan Company's interactions in a mock banking system

Alphabet
(Use LTSA's compressed notation, if alphabet is large.)
	c.{balance.{{acquire, no_funds}, read[0..1], release, write[0..2]}, give_loan}
	
	p.{balance.{{acquire, no_funds}, read[0..1], release, write[0..2]}, send_money}
	
	s.{balance.{{acquire, no_funds}, read[0..1], release, write[0..1]}, buy_iPhone}
	
	u.{balance.{{acquire, no_funds}, read[0..1], release, write[0..1]}, take_fee}}

Sub-processes (List them.)	ACCOUNT, STUDENT, PARENT, UNIVERSITY, COMPANY, LOCK, LOCKED
Number of States	42
Deadlocks (yes/no)	no
Deadlock Trace(s)	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 sub-processes.)

FSP Program:

```
const Max = 1
range M = 0..Max

set AccountAlpha = { balance.{read[M], write[M], acquire, release,
no_funds} }

ACCOUNT = ACCOUNT[0],

ACCOUNT[x : M] =    ( read[x] -> ACCOUNT[x] | write[money : M] ->
ACCOUNT[money] ) .



LOCK = ( acquire -> release -> LOCK ) .

|| LOCKED = ( LOCK || ACCOUNT ) .




||   BANKING_SYSTEM
=(   s: STUDENT
||   u: UNIVERSITY
||   p: PARENT
||   c: COMPANY
||   {s,u,c,p} :: balance : LOCKED ) .
```

3. Combined Sub-processes

(Add rows as necessary.)

Process	Description
STUDENT	Represents the student who will buy an iPhone if the funds are available in their account
UNIVERSITY	Represents the university who will take the student's fees if the funds are available
PARENT	Represents the parent of the student who will add funds to their account
COMPANY	Represents the loan company who will add funds by issuing the students loan
ACCOUNT	Represents the actual physical account and possible actions
LOCK	Represents the fact that only one process can access the account at a given time

4. Analysis of Combined Process Actions

- **Synchronous** actions are performed by at least two sub-process in the combination.
- **Blocked Synchronous** actions cannot be performed, since at least one of the sub-processes cannot perform them, because they were added to their alphabet using alphabet extension.
- **Asynchronous** actions are preformed independently by a single sub-process.

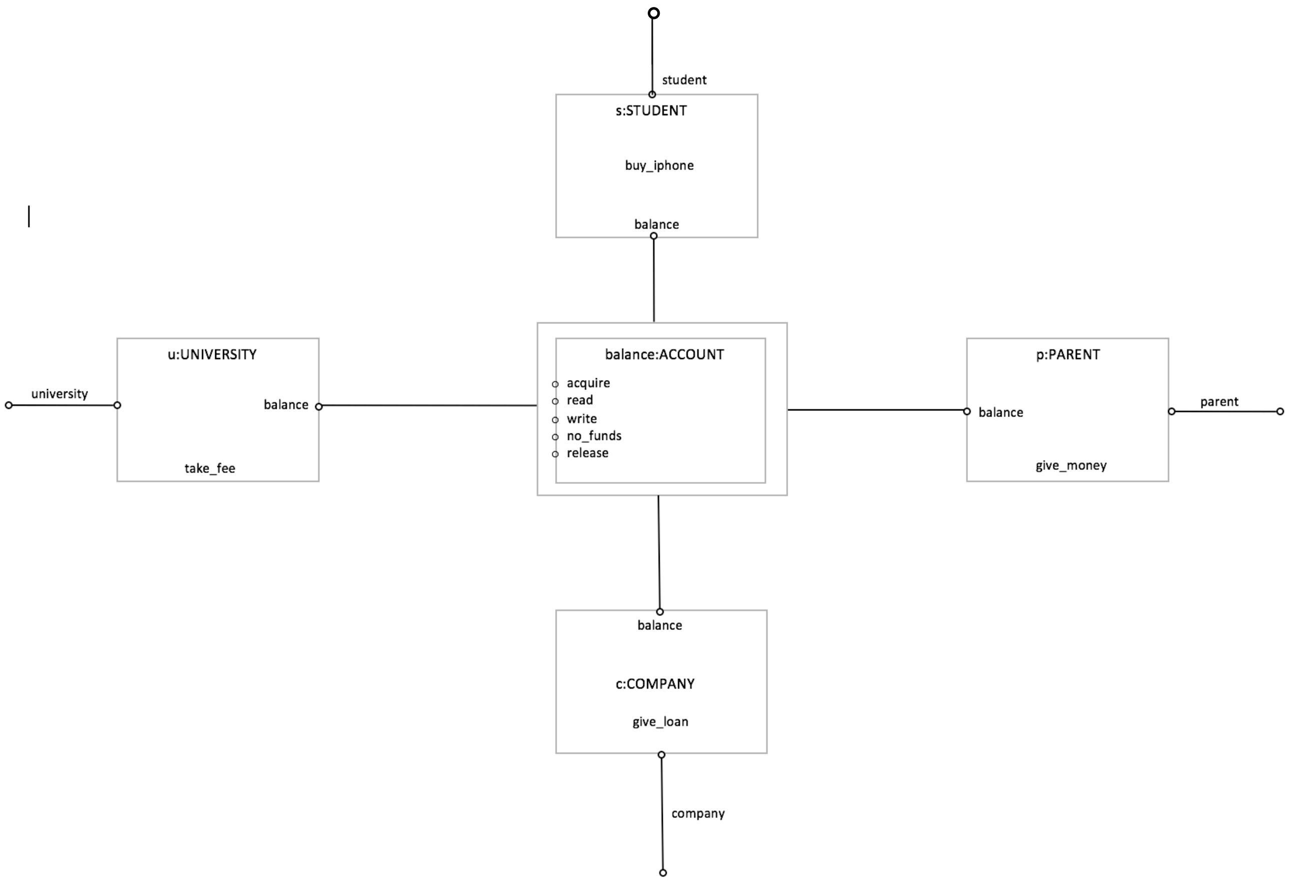
(Add rows as necessary.)

Synchronous Actions	Synchronised by Sub-Processes (List)
balance.read[x]	ACCOUNT, STUDENT, PARENT, COMPANY, UNIVERSITY
balance.write[x]	ACCOUNT, STUDENT, PARENT, COMPANY, UNIVERSITY
balance.acquire	STUDENT, PARENT, COMPANY, UNIVERSITY
balance.release	STUDENT, PARENT, COMPANY, UNIVERSITY
balance.no_funds	STUDENT, UNIVERSITY

Sub-Process	Asynchronous Actions (List)
STUDENT	buy_iPhone
PARENT	send_money
UNIVERSITY	take_fee
COMPANY	give_loan

5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.



student
s:STUDENT
buy_iphone
balance
u:UNIVERSITY
university
balance
take_fee
balance:ACCOUNT
acquire
read
write
no_funds
release
p:PARENT
balance
parent
give_money
balance
c:COMPANY
give_loan
company