Tutorial 10

1) Let [BORROWER] be the collection of all possible library borrowers, and [COPY] be the collection of all possible copies of items that may be loaned to borrowers.

There is a limit to the number of copy that can be loaned to borrowers:

| lendingLimit : ℕ

A state space for a *Library* is given by the following schema:

_Library____

borrower: \mathbb{F} BORROWER

 $copy : \mathbb{F} COPY$

loan : *COPY* → *BORROWER*

 $dom\ loan \subseteq copy$

 $ran\ loan\ \subseteq\ borrower$

 $loan = \{c : COPY; b : BORROWER \mid c \mapsto b \in loan \land \#(loan \triangleright b) \leq lendingLimit\}$

Given the specification of error handling as below:

REPORT ::= okay | notAMember | lendingLimitReached | notALoanCopy | copyOutOnLoan

(a) Rewrite the specification for *Issue* below using Z schema.

Use case:	Issue
Purpose:	To loan a copy of a book to a borrower.
Pre-conditions:	The copy is a part of the copy to be loaned. The borrower is a member of the library. The borrower has not reached his lending limit The copy is not out on loan.
Initiating actor:	Librarian
Main success Scenario	 Librarian inputs borrower Librarian inputs copy System confirms copy is loaned to borrower Exit success
Exceptions	 1a) Borrower is not a member of the library 1a1 Exit failure 1b) Borrower has reached maximum number of loans allowed 1b1 Exit failure 2a) Copy is not a part of the copy to be loaned 2a1 Exit failure

2b) Copy is out on loan
2b1 Exit failure

Issue

```
\Delta Library
b?:BORROWER
c?:COPY

c? ⊆ copy
c? ∉ dom loan
b? ⊆ borrower
\#(loan ▷ b?) < lendingLimit

copy' = copy
borrower' = borrower
loan' = loan ∪ {c? ▷ b?}
```

(b) Write a schema called *Success* that will provide a response for every successful error handling schema.

Success

```
rep! : REPORT

rep! = okay
```

(c) Referring to the specification described in (a) above, rewrite the exception in Z schemas called *IssueError*.

NotALoanCopy

```
E Library
b?: BORROWER
rep!: REPORT

c? ∉ copy
rep! = notALoanCopy
```

```
AlreadyOutOnLoan
Ξ Library
c? : COPY
rep! : REPORT
c? \in dom \ loan
rep! = AlreadyOutOnLoan
NotAMember  
Ξ Library
b? : BORROWER
rep!: REPORT
b? ∉ borrower
rep! = notAMember
LimitReached
Ξ Library
b? : BORROWER
rep!: REPORT
\#(loan \triangleright b?) \ge lendingLimit
rep! = lendingLimitReached
OR
LimitReached _____
Ξ Library
b?: BORROWER
rep!: REPORT
(c? \notin copy \land rep! = notALoanCopy)
(c? \in dom\ loan \land rep! = AlreadyOutOnLoan)
(b? \notin borrower \land rep! = notAMember)
(\#(loan \triangleright b?) \ge lendingLimit \land rep! = lendingLimitReached)
```

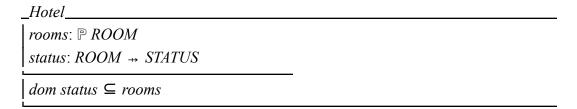
(d) Define the complete operation called *CompleteIssue* which caters all possible violation of the precondition in the schema *Issue*.

```
CompleteIssue \hat{} (Issue \land Success) \lor NotALoanCopy \lor AlreadyOutOnLoan \lor NotAMember \lor LimitReached
```

2) Let [ROOM] be the set of all possible hotel rooms. The hotel has a set of rooms named *rooms* and each room is either occupied or vacant.

```
STATUS ::= vacant | occupied
```

When a new room is added, it is always vacant. A specification for a hotel has a state space schema:



- (a) Referring to the state space above, introduce a free type for an exception handling message called *REPORT* that has four errors handling. The error handling messages are *success*, *alreadyAdded*, *notInSystem*, and *roomOccupied*.
- (b) Write an error handling schema called *DuplicateRoom* where we cannot add the same room twice into our system.
- (c) Write an error handling schema called *NoSuchRoom* where we cannot place a guest in a room, or remove the room from the system, if it does not exist.
- (d) Write an error handling schema called *RoomNotEmpty* where we cannot place a guest in a room, or remove the room from the system, if it is already occupied.
- (e) Define the total operation for occupying a room as *CompleteOccupy* which caters possible violation of the precondition for the schema *OccupyRoom*.
- (f) Define the total operation for adding a room as *CompleteAdd* which caters possible violation of the precondition for the schema *AddRoom*.