Tutorial 7

1) Compare a *relation* with a *function*.

Relation

e.g. publicHoliday : COUNTRY $\leftrightarrow$ DATE

- many-to-many

Function

e.g. staffName : STAFFID ⇸ NAME
staffAge : STAFFID ⇸ $\mathbb{N}$

- many-to-many or one-to-one

2) What is the difference between a *partial* function and a *total* function.

⇸ partial function

e.g. staffName : STAFFID ⇸ NAME

$\rightarrow$ total function

e.g. hasAge : PERSON $\rightarrow \mathbb{N}$
hasMother : PERSON $\rightarrow$ PERSON
hasCapital : COUNTRY $\rightarrow$ CITY

Both can be many-to-one or one-to-one

3) Referring to Tutorial 3 question 2,

BankAccount

active, overdrawn, depositor, current : $\mathbb{P}$ ACCNO

overdrawn $\subseteq$ active
overdrawn $\subseteq$ current
depositor $\cup$ current = active
depositor $\cap$ current = $\varnothing$
depositor $\cap$ overdrawn = $\varnothing$

(a) Introduce *balance*, the balance in each account where it is recorded by a partial function between account numbers and integers (used to record sums of money)

balance : ACCNO ⇸ $\mathbb{Z}$

(b) Write down predicates to formalize the following statements:

(i) The active accounts are those for which there are balances

dom balance = active
OR
active = dom balance

(ii) Overdrawn accounts are those accounts that have negative balances

$overdrawn = \{acc : ACCNO \mid acc \in \mathrm{dom}\ balance \land balance(acc) < 0\}$

OR

$overdrawn = \{acc : \mathrm{dom}\ balance \mid balance(acc) < 0\}$

OR

$overdrawn = \mathrm{dom}(balance \rhd \{bal : \mathbb{Z} \mid bal < 0\})$

(c) Write down an expression for the set of account numbers of deposit accounts with balances exceeding 100000.

$\{acc : \mathrm{dom}\ balance \mid balance(acc) > 100000\} \cap depositor$

OR

$\mathrm{dom}\ (balance \rhd \{bal : \mathbb{Z} \mid bal > 100000\}) \cap depositor$

(d) The bank keeps details of its customer, and *CUSTOMER* is the given set of customer details:

[CUSTOMER]
details : ACCNO ⇸ CUSTOMER

Formalize the rule that the active accounts and those for which there are customers' details are the same.

$active = \mathrm{dom}\ details$

OR

$\mathrm{dom}\ details = active$

(e) If *c:CUSTOMER* is a particular customer, write down an expression for the account numbers associated with *c*.

$\mathrm{dom}\ (details \rhd \{c\})$

OR

$detail{\sim}\llparenthesis\{c\}\rrparenthesis$

4) Consider a system description of the *Project Allocation* system, as provided below:

Each programmer employed within the organisation is uniquely identified by a programmers' IDs. The name of any programmer may be derived from their programmers' IDs. Each project is uniquely identified by a project code from which the project name may be derived.

Given the basic types:

[PROGID]	- the set of all possible programmers' IDs
[PROGNAME]	- the set of all possible programmers' names
[PROJCODE]	- the set of all possible project codes
[PROJNAME]	- the set of all possible project names

Construct a state space schema called *ProjectAllocation* that will specify the following properties of the system:

- *programmer* which describes the connection between programmers' IDs and programmers' names
- *project* which describes the connection between project codes and project names
- *assignment* which describes the connection between programmers' IDs and project codes.

You must include all possible invariants in the state space schema.

ProjectAllocation

$programmer : PROGID \nrightarrow PROGNAME$
$project : PROJCODE ⤔ PROJNAME$
$assignment : PROGID \leftrightarrow PROJCODE$

$\mathrm{dom}\ programmer = \mathrm{dom}\ assignment$
$\mathrm{dom}\ project = \mathrm{ran}\ assignment$

5) Interpret each line (1) to (6) of Z schemas below using appropriate meaning in plain English language.

[ROOM]
STATUS ::= occupied | vacant (1)

Hotel

$room : ROOM ⇻ STATUS$ (2)

Occupy

$\Delta\ Hotel$
$room? : ROOM$ (3)

$room? \in \mathrm{dom}\ room$ (4)
$room\ (room?) = vacant$ (5)
$room' = room \oplus \{room \mapsto occupied\}$ (6)

(1) Room status can be occupied or vacant.
(2) ⇻ finite function : In a hotel , there is a finite number of room, an each room has status.
(3) To occupy the room, you need to get the input of the room.
(4) The room to be occupied, must be exist in the hotel.
(5) The room to be occupied must be vacant / The status of the room to be occupied must be vacant.
(6) Update the room statue from vacant to occupied.