Tutorial 11

1) Given [CAR] as the set of all possible cars in a system, interpret these variable declarations:

car1: $\mathbb{P}$ CAR

- set, data cannot be duplicated. Data does not need to be in order
- E.g. car1 == {perodua, proton, honda, toyota}

car2: seq CAR

- sequence, data may be duplicated. Data needs to be in order
- E.g. car2 == ⟨perodua, proton, honda, toyota⟩
  or car2 == {1 ↦ perodua, 2 ↦ proton, 3 ↦ honda, 4 ↦ toyota,
  5 ↦ proton}

car3: bag CAR

- bag, data may be duplicated but does not need to be in order
- E.g. car3 == [[perodua, proton, honda, toyota]]
  or car3 == {perodua ↦ 1, proton ↦ 2, honda ↦ 1, toyota ↦ 1}

2) Suppose s = ⟨3, 7, 1, 2⟩ and t = ⟨5, 9⟩, indicate the result for the following statements:

(a) head s
⟨3⟩

(b) tail s
⟨7, 1, 2⟩

(c) last s
⟨2⟩

(d) front s
⟨3, 7, 1⟩

(e) $s \frown t$
⟨3, 7, 1, 2, 5, 9⟩

(f) $\langle 8 \rangle \frown s$
⟨8, 3, 7, 1, 2⟩

3) Let [BOOK] be the set of all possible books in TARUMT. The TARUMT library catalogue could be of type *bag BOOK*.

tarumt : bag BOOK

Given tarumt = ⟦Spivey, Spivey, Diller, Peled⟧, indicate the result for the following statements:

(a) tarumt # Spivey
2

(b) count tarumt Woodcock
()

(c) 2 ⊗ tarumt
⟦Spivey, Spivey, Diller, Peled, Spivey, Spivey, Diller, Peled⟧

(d) tarumt ⩀ ⟦Spivey, Spivey⟧
⟦Diller, Peled⟧

(e) tarumt ⊎ ⟦Spivey, Spivey⟧
⟦Spivey, Spivey, Diller, Peled, Spivey, Spivey⟧

4) Write the output for each of the following expressions:

(a) ⟨r, o, v, e, r⟩ ↾ ⟨o, w, l⟩
⟨o⟩

(b) ⟨o, w, n⟩ ⁀ ⟨e, r, s⟩
⟨o, w, n, e, r, s⟩

(c) ⟨p⟩ ⁀ ⟨r, o, v, e, r⟩
⟨p, r, o, v, e, r⟩

(d) tail ⟨c, h, e, c, k, e, r⟩
⟨h, e, c, k, e, r⟩

(e) ⟦Book, Pen, Book, Ruler, Pen, Book, Eraser⟧ # ⟦Ruler⟧
1

(f) 2 ⊗ ⟦f, m, s, e⟧
⟦f, m, s, e, f, m, s, e⟧

(g) ⟦Book, Pen, Book, Ruler, Pen, Book, Eraser⟧ ⩀ ⟦Book, Ruler⟧
⟦Pen, Book, Pen, Book, Eraser⟧

(h) squash {3↦g, 1↦t, 2↦e, 4↦r, 5↦k, 7↦p, 6↦n}
⟨t, e, g, r, k, n, p⟩

(i) squash $\{3 \mapsto g, 10 \mapsto t, 2 \mapsto e, 14 \mapsto r, 5 \mapsto k, 12 \mapsto p, 6 \mapsto n\}$
$\langle e, g, k, n, t, p, 2 \rangle$

5) Consider a scenario where you are going to model a toll plaza system concerning queuing of vehicles at the toll booths. Suppose you are given two types:

[BOOTH] - the set of all possible booths at the toll plaza
[VEHICLE] - the set of all possible vehicles queuing at the toll booths

The state of the *TollPlaza*, at any time, can be expressed in the Z state space below:

TollPlaza

$booth : \mathbb{P}\ BOOTH$
$queue : BOOTH \pfun iseq\ VEHICLE$

$dom\ queue = booth$

Referring to the given state space schema for the *TollPlaza* above, construct the Z schemas for the following operations:

(a) An initial schema called *InitToll* which defines the initial condition of each of the components.

InitToll

TollPlaza

$booth = \varnothing$

(b) An operation schema called *JoinQueue* such that a vehicle *v?* joins a queue at a toll booth *b?*.

JoinQueue

$\Delta\ TollPlaza$
$v? : VEHICLE$
$b? : BOOTH$

$b? \in booth$ (exist in the system)
$b? \in dom\ queue$ (open for queuing)
$v? \notin ran\ (queue\ b?)$

$queue' = queue \oplus \{b? \mapsto queue\ b? \frown \langle v? \rangle\}$
$booth' = booth$

queue b? = iseq VEHICLE
ran (queue b?) = VEHICLE

(c) An operation schema called *LeaveQueue* such that a vehicle *v?* leaves a queue at a toll booth *b?* after payment is made.

__*LeaveQueue*__

Δ *TollPlaza*
$v? : VEHICLE$
$b? : BOOTH$

$b? \in booth$ (exist in the system)
$b? \in \mathrm{dom}\ queue$ (open for queuing)
$v? \in \mathrm{ran}\ (queue\ b?)$
$v? = head\ (queue\ b?)$

$queue' = queue \oplus \{b? \mapsto tail\ (queue\ b?)\}$
$booth' = booth$

(d) An operation schema called *TotalVehicle* that will provide the total number of vehicles currently queuing at a toll booth *b?*.

__*TotalVehicle*__

Ξ *TollPlaza*
$b? : BOOTH$
$total! : \mathbb{N}$

$b? \in booth$ (exist in the system)
$b? \in \mathrm{dom}\ queue$ (open for queuing)
$total! = \#(queue\ b?)$

$queue' = queue$
$booth' = booth$
