Concordia University Department of Computer Science and Software Engineering

SOEN 331 Section S: Formal Methods

for Software Engineering

Assignment 2

Mohammad Ali Zahir - 40077619

Marwa Khalid - 40155098

October 17, 2022

Date of Submission: October 31, 2022

Contents

1   System Requirements

Consider a system such as *flightradar24.com*. A flight is associated with **a flight number** (such as UA79), a specific code that an airline assigns to a particular flight in its network, and **route** which is a source-destination city pair such as *(NY ,Tokyo)*. For example, the United Airlines flight from New to Tokyo is tracked by the system as $UA79 \mapsto (NY, Tokyo)$. The formal specification of the system introduces the following three types:

> *FLIGHT_NUMBER*,
> *ROUTE*,
> *CITY*

where

$ROUTE : CITY \times CITY$

Flight numbers are unique, and there are possibly several flights that cover the same route. For example, there are possibly several flights from New York to Tokyo. The system must keep track of all active flights. Formally, let us have the following variables:

1. *active*: holds all active flight numbers.

2. *map*: holds a collection of active flight-route pairs.

2   Your Assignment

1. (2 pts) Provide a declaration for variable *active*.

   <u>Solution:</u>

   The declaration of the of the variable active would be: (Assuming all flight numbers that we keep track of are all active flight numbers)

   $Active : \mathbb{P} \; FLIGHT\_NUMBER$

2. (3 pts) What kind of collection is variable *map*.

   Solution:

   For this collection, it mentions that we have pairs of two of the 3 types, we can have multiple flights going to the same route. It also means that we have a pair of 2-elements here, which is called an ordered pair, **which would mean the map collection is a tuple**.

3. (10 pts) Is variable map a function and if so, comment on whether it is a total or partial function, as well as on the properties of injectivity, surjectivity and bijectivity?

   Solution:

   Since we would say the map always has to be linked to an active flight-route pair, it would mean that we would have a total function here. In terms of what type of property this function, this cannot be a interjective function since we may have multiple flights going to the same route, hence we do not have a one-to-one function. For bijection, the function needs to be both interjective and surjective, and since we have proved that the function is not interjective, it means that it is not bijective as well. **Hence, the variable map is a total surjective function.**

4. (10 pts) Provide a formal specification of the state of the system in terms of a **Z specification schema**.

   Solution:

$$
\begin{array}{|l}
\underline{\quad flightradar24.com\ \rule[0pt]{6cm}{0.4pt}} \\
active\ :\ \mathbb{P}\ FLIGHT\_NUMBER \\
map : FLIGHT\_NUMBER \rightarrow ROUTE \qquad \texttt{--total surjective} \\
\hline
active = \mathrm{dom}\ map \\
\end{array}
$$

5. (15 pts) Provide a schema for operation *RegisterFlightOK* that adds a flight to the tracker. With the aid of success and error schema(s), provide a definition for operation *RegisterFlight* that the system will place in its exposed interface.

Solution:

For question 5, 6, 7, to create the error and success schemas, we will create a new enumrated value which will output success and error which we call RESULT.

RegisterFlightOK
Δ _flightradar24.com_
_flight_number?_ : _FLIGHT_NUMBER_
route! : _ROUTE_

_flight_number?_ $\notin$ _active_
route? $\notin$ ran _map_
active' = _active_ $\cup$ {_flight_number?_}
map' = _map_ $\cup$ {_flight_number?_ $\mapsto$ _route?_}

Success
Ξ _flightradar24.com_
response! : _RESULT_

---
response! = _'SUCCESS'_

ActiveFlightExists
Ξ _flightradar24.com_
_flight_number?_ : _FLIGHT_NUMBER_
route? : _ROUTE_
response! : _RESULT_

_flight_number?_ $\in$ _active_
response! = _'Flight having flight_number? is already active'_

$RegisterFlight \ \hat{=}$
$\quad (RegisterFlightOK \wedge Success) \oplus ActiveFlightExists$

6. (15 pts) Provide a schema for operation _GetRouteOK_ that returns the route given its flight. With the aid of success and error schema(s), provide a definition for operation _GetRoute_ that the system will place in its exposed interface.

Solution:

GetRouteOK

$\Xi\ flightradar24.com$
$flight\_number? : FLIGHT\_NUMBER$
$route! : ROUTE$

$flight\_number? \in active$
$route! = map(flight\_number?)$

Success

$\Xi\ flightradar24.com$
$response! : RESULT$

$response! = \ 'SUCCESS'$

RouteNotFound

$\Xi\ flightradar24.com$
$flight\_number? : FLIGHT\_NUMBER$
$response! : RESULT$

$flight\_number? \notin active$
$route? \notin map$
$response! = \ 'Since\ flight\_number?\ is\ not\ active\ , unable\ to\ find\ route'$

$GetRoute \ \widehat{=}$
$\quad (GetRouteOK \land Success) \oplus RouteNotFound$

7. Provide a schema for operation *GetFlightOK* that returns any and all active flights given a route. With the aid of success and error schema(s), provide a definition for operation *GetFlight* that the system will place in its exposed interface.

Solution:

```
┌─ GetFlightOK ────────────────────────────────────
│ Ξ flightradar24.com
│ flight_number! : FLIGHT_NUMBER
│ route? : ROUTE
├──────────────────────────────────────────────────
│ route? ∈ ran map
│ flight_number! = map(route?)
└──────────────────────────────────────────────────
```

```
┌─ Success ────────────────────────────────────────
│ Ξ flightradar24.com
│ response! : RESULT
├──────────────────────────────────────────────────
│ response!  =  'SUCCESS'
└──────────────────────────────────────────────────
```

```
┌─ FlightNotFound ─────────────────────────────────
│ Ξ flightradar24.com
│ route? : ROUTE
├──────────────────────────────────────────────────
│ route? ∉ ran map
│ response!  =  'Unable to find flight with this route?'
└──────────────────────────────────────────────────
```

$GetFlight \; \hat{=}$
  $(GetFlightOK \land Success) \oplus FlightNotFound$