SOEN 331: Introduction to Formal Methods for Software Engineering

Assignment 2

The Object-Z specification language

Duc Nguyen - 40064649
Vithura Muthiah - 40062305
Auvigoo Ahmed - 40128901
Ali Hanni - 40157164

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Winter 2021

Contents

1 Problem 1: State (7 pts)

1.1 Description:

The declaration of the state of the system is defined by

- The set of phone numbers (call it *numbers*) that are recorded in contacts
- A record of association between names and phone numbers, given by a correspondence (call it *recorded*).

1. Provide a diagram to visualize the state of the system.
2. Provide a formal definition for numbers.
3. Does *recorded* have to be captured by a function? What requirements would a function enforce? Explain in detail.
4. What is the domain and the codomain of *recorded*?
5. What type of function should *recorded* be (full or partial)? Explain in detail.
6. Will *recorded* be an injective, surjective, or bijective? Explain in detail.
7. Provide a formal definition for *recorded*.

1.2 Answer:

1. The following figure visualizes the state of the system:

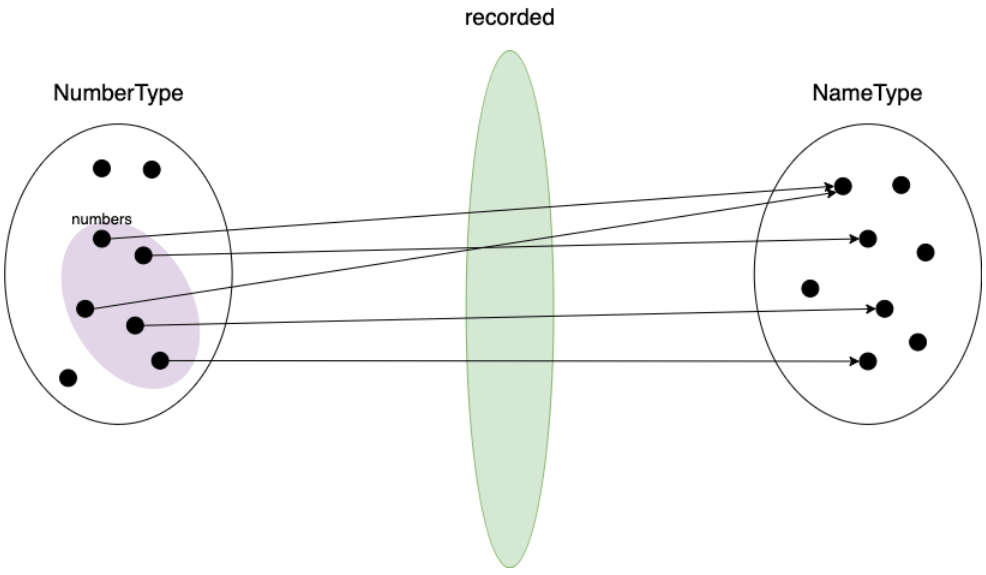

Figure 1: State of the System

2. *numbers* can be formally defined as:
   $\{\forall\, x, y : numbers \mid (x \in numbers \land y \in numbers) \rightarrow x \neq y\}$

3. *recorded* has to be a function since it is mapping 2 sets (*numbers* and *NameType*) and every member in *numbers* has to be associated to exactly one name in *NameType*.
   A function requires 2 sets A and B present and there exists assignments from elements in A to elements in B.

4. - Domain of *recorded* is *numbers* since it is the set contains elements that can be used as input to function *recorded*.
   - Codomain of *recorded* is *NameType* since it is a set of elements that can possibly be derived from function *recorded*.

5. As previously stated, the domain of *recorded* is *numbers*. We know that *numbers* is the set of all phone numbers recorded in contacts. Since not all elements of *PhoneNumberType* are recorded, we can state that *numbers* is a subset of *PhoneNumberType* as shown in the diagram displayed at question 1. A partial function f is a function that is defined for some subset A' of A, not forcing mapping for all elements of set A such that $dom\ f \subset A$. Thus, *recorded* is a partial function.

6. - Given the definition of an injective function $(\forall\, a, b \mid a \neq b \rightarrow f(a) \neq f(b))$, since multiple elements of *numbers* can be associated with a single element of *NameType*, then recorded is not injective.
   - Also, given the definition of a surjective function $(\forall\, b, \exists\, a \mid f(a) = b)$, since not all elements of *NameType* are mapped to by at least one element of *numbers*, *recorded* is also not a surjective function.
   - Since it is neither injective nor surjective, recorded cannot be bijective.

     $\rightarrow$ Therefore, *recorded* is not one-to-one and not onto.

7. The function *recorded* can be formally defined as:
   $\{\exists\, y \in NameType, \exists\, x \in numbers \mid recorded(x) = y\}$

2 Problem 2: Class Contacts (35 pts)

2.1 Description:

Define a formal specification in Object-Z for class *Contacts* whose interface contains the following *robust specifications*:

- **MakeNewContact**: Adds a new person to Contacts with a single phone number.
- **AddNumber**: Adds an additional phone number for an existing contact.
- **SearchForNumber**: : Returns a collection of phone numbers for a given person.
- **DeleteNumber**: Deletes an existing number.

2.2 Answer:

Contacts

$\upharpoonright (MakeNewContact, AddNumber, SearchForNumber, DeleteNumber)$

$$numbers : \mathbb{P}\, PhoneNumberType$$
$$recorded : PhoneNumberType \pfun NameType$$
$$\hline$$
$$numbers = dom\ recorded$$

Init

$$recorded = \varnothing$$

MakeNewContactOK

$$\Delta(recorded, numbers)$$
$$number? : PhoneNumberType$$
$$name? : NameType$$
$$\hline$$
$$number? \notin numbers$$
$$name? \notin ran\ recorded$$
$$recorded' = recorded \cup \{number? \mapsto name?\}$$
$$numbers' = numbers \cup \{number?\}$$

AddNumberOK

$$\Delta(recorded, numbers)$$
$$number? : PhoneNumberType$$
$$name? : NameType$$
$$\hline$$
$$number? \notin numbers$$
$$name? \in ran\ recorded$$
$$recorded' = recorded \cup \{number? \mapsto name?\}$$
$$numbers' = numbers \cup \{number?\}$$

SearchForNumberOK

$$\Xi(recorded)$$
$$name? : NameType$$
$$number! : \mathbb{P}\, PhoneNumberType$$
$$\hline$$
$$name? \in ran\ recorded$$
$$number! = \{x : numbers \mid recorded(x) = name?\}$$

DeleteNumberOK

$$\Delta(recorded, numbers)$$
$$number? : PhoneNumberType$$
$$\hline$$
$$number? \in numbers$$
$$recorded' = \{number?\} \ndres recorded$$
$$numbers' = numbers \setminus \{number?\}$$

Success

$response! : Message$

$response! = \text{`}Success\text{'}$

NumberExists

$number? : PhoneNumberType$
$response! : Message$

$number? \in numbers$
$response! = \text{`}Number\ already\ exists\text{'}$

NumberNotFound

$number? : PhoneNumberType$
$response! : Message$

$number? \notin numbers$
$response! = \text{`}Number\ not\ found\text{'}$

NameExists

$name? : NameType$
$response! : Message$

$name? \in ran\ recorded$
$response! = \text{`}Name\ already\ taken\text{'}$

NameNotFound

$name? : NameType$
$response! : Message$

$name? \notin ran\ recorded$
$response! = \text{`}Name\ not\ found\text{'}$

$MakeNewContact \mathrel{\widehat=} (MakeNewContactOK \land Success) \oplus (NumberExists \lor NameExists)$
$AddNumber \mathrel{\widehat=} (AddNumberOK \land Success) \oplus (NumberExists \lor NameNotFound)$
$SearchForNumber \mathrel{\widehat=} (SearchForNumberOK \land Success) \oplus NameNotFound$
$DeleteNumber \mathrel{\widehat=} (DeleteNumberOK \land Success) \oplus NumberNotFound$

3 Problem 3: Class Contacts2 (8 pts)

3.1 Description:

Subclassify **Contacts** to introduce class **Contacts2** that behaves exactly like **Contacts**, while introducing a robust operation to search for a person, given a phone number through operation **SearchForPerson**.

3.2 Answer:

Contacts2

$\upharpoonright (MakeNewContact, AddNumber, SearchForNumber, DeleteNumber, SearchForPerson)$

$Contacts$

SearchForPersonOK

$\Xi(recorded)$

$number? : PhoneNumberType$

$name! : NameType$

$number? \in numbers$

$name! = recorded(number?)$

$SearchForPerson \mathrel{\widehat=} (SearchForPersonOK \land Success) \oplus NumberNotFound$