Concordia University

Department of Computer Science and Software Engineering

SOEN 331 - S and U

Introduction to Formal Methods for Software Engineering

Assignment 2 - Solutions

The Object-Z specification language

Team 19 - Section U

Samuel Boaknin 40009692

Ryan Leyland 40015165

Saleha Tariq 40006997

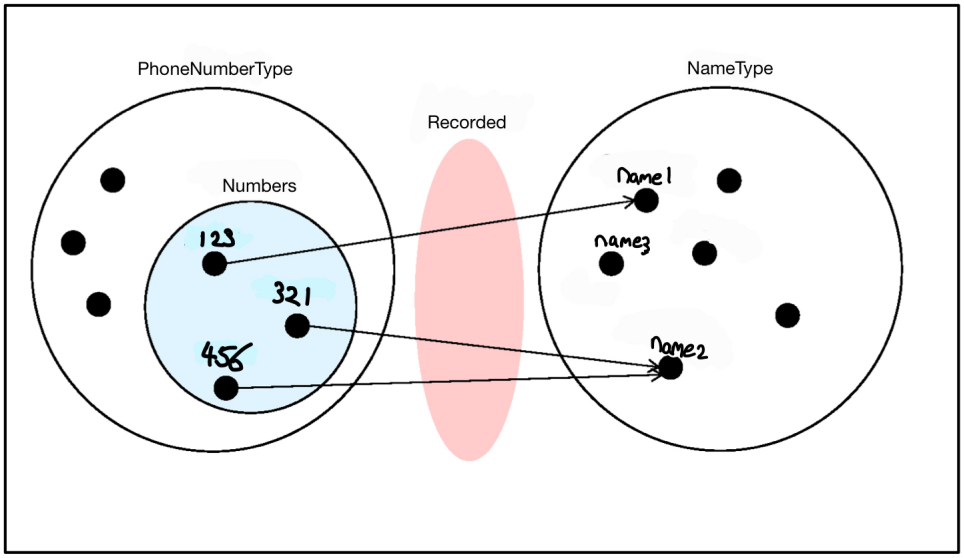
Meng Susana Ung 40099729

Due Date: March 11, 2021

1 Contact management

1.1 State

1. This is our diagram for the state of the system:



2. The formal definition of *numbers* is as follows:

$$numbers \ : \ \mathbb{P} \ PhoneNumberType$$

3. The relation *recorded* must indeed be captured by a function. A function, by definition, enforces that each element of a source set (that is, the set in the domain) is mapped to exactly one element of a target set (a set of elements in the range). Part of the requirements for the system is that people may not share phone numbers. By making *recorded* a function, and setting the *numbers* set as a subset of the domain, the requirement of a phone number being associated to a single person is trivially met. If this is not done, an additional series of specifications must be defined such that this requirement is observed *at best*, and at worst, there is a severe risk of failing to meet the defined requirements. Thus, it is at least convenient, if not necessary that *recorded* is a function where *numbers* is a subset of the domain.

4. The domain of *recorded* is *numbers*, and its codomain is *NameType*.

5. The function *recorded* is a partial function, because partial functions, by definition, do not map all elements in the domain to an element in the co-domain. Specifically, *recorded* doesn't relate every element of PhoneNumberType to NameType.

6. The function *recorded* will not be either injective, surjective, or bijective. It cannot be injective because an injective function, by definition, is a function where one element in the domain is mapped to at most one element in the co-domain, but recorded allows for multiple phone numbers to be associated with one person, so multiple parts of the domain could be associated with an element of the co-domain. It cannot be surjective as a surjective function, by definition, is a function where each element in the co-domain has at least one element in the domain mapped to it. In *recorded*, there are elements contained in the co-domain of *recorded* that are not in the range of *recorded* therefore there exists elements in the codomain which are not mapped to by any element of the domain. It cannot be bijective because a bijective function is both injective and surjective and it has already been proven that this function is neither. Therefore, *recorded* must be a general function because every element in the domain is mapped to one element of the co-domain, but more than one element of the domain can be mapped to the same element of the co-domain.

7. The formal definition of *recorded* is as follows:

$$recorded \ : \ PhoneNumberType \nrightarrow NameType$$

1.2 Class Contacts

Contacts

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

$$
\begin{array}{l}
numbers : \mathbb{P}\, PhoneNumberType \\
recorded : PhoneNumberType \pfun NameType \\
count : \mathbb{N} \\
\hline
numbers = dom\ recorded \\
count >= 0
\end{array}
$$

Init

$$
\begin{array}{l}
recorded = \varnothing \\
count = 0
\end{array}
$$

MakeNewContactOK

$$
\begin{array}{l}
\Delta(recorded, count) \\
number? : PhoneNumberType \\
name? : NameType \\
\hline
number? \notin numbers \\
name? \notin ran\ recorded \\
recorded' = recorded \cup \{number? \mapsto name?\} \\
count' = count + 1
\end{array}
$$

AddNumberOK

$$
\begin{array}{l}
\Delta(recorded) \\
number? : PhoneNumberType \\
name? : NameType \\
\hline
count > 0 \\
number? \notin numbers \\
name? \in ran\ recorded \\
recorded' = recorded \cup \{number? \mapsto name?\}
\end{array}
$$

SearchForNumberOK

$$
\begin{array}{l}
\Xi Contacts \\
name? : NameType \\
numbers! : \mathbb{P}\, number \\
\hline
count > 0 \\
numbers! = \{n : number \mid recorded(n) = name?\}
\end{array}
$$

...

Contacts/cont.

...

DeleteNumberOK

$\Delta(recorded, count)$
$number? : PhoneNumberType$
$name : NameType$

$count > 0$
$name = recorded(number?)$
$number? \in numbers$
$recorded' = \{number?\} \lhd recorded$
$\textbf{if } \#(recorded \rhd \{name\}) = 0 \textbf{ then } count' = count - 1 \textbf{ else } count' = count$

NameUnknown

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

$name? \notin ran\ recorded$
$response! = \ 'Name\ unknown'$

NumberUnknown

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

$number? \notin dom\ recorded$
$response! = \ 'Number\ unknown'$

NameExists

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

$name? \in ran\ recorded$
$response! = \ 'Name\ already\ exists'$

NumberExists

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

$number? \in dom\ recorded$
$response! = \ 'Number\ already\ exists'$

ContactsEmpty

$response! : Message$

$count = \ 0$
$response! = \ 'Contacts\ is\ empty'$

Success

$response! : Message$

$response! = \ 'Success'$

...

Contacts/cont.

...

$MakeNewContact \mathrel{\widehat{=}} (MakeNewContactOK \land Success) \oplus (NameExists \lor NumberExists)$

$AddNumber \mathrel{\widehat{=}} (AddNumberOK \land Success) \oplus (NameUnknown \lor NumberExists \lor ContactsEmpty)$

$SearchForNumber \mathrel{\widehat{=}} (SearchForNumberOK \land Success) \oplus (NameUnknown \lor ContactsEmpty)$

$DeleteNumber \mathrel{\widehat{=}} (DeleteNumberOK \land Success) \oplus (NumberUnknown \lor ContactsEmpty)$

1.3 Class Contacts2