Assignment 2-sol - a2solution

Introduction to Formal Methods for Software Engineering (Concordia University)

Concordia University

Department of Computer Science and Software Engineering

SOEN 331-S:

Formal Methods for Software Engineering

Solutions to Assignment 2 on

Relational calculus and Z specification

Dr. Constantinos Constantinides, P.Eng.

constantinos.constantinides@concordia.ca

October 27, 2023

Contents

1 General information

Date posted: Friday 27 October, 2023.

Date due: Friday, 10 November, 2023, by 23:59.

Weight: 10% of the overall grade.

2 Introduction

You should find one partner and between the two of you should designate a team leader who will submit the assignment electronically. There are **2** problems in this assignment, with a total weight of **100** points.

3 Ground rules

1. This is an assessment exercise. You may not seek any assistance while expecting to receive credit. **You must work strictly within your team and seek no assistance for this assignment (e.g. from the teaching assistants, fellow classmates and other teams or external help)**. You should **not** discuss the assignment during tutorials. I am available to discuss clarifications in case you need any.

2. **Both partners are expected to work relatively equally on each problem**. Accommodating a partner who did not contribute will result in a penalty to **both**. You cannot give a "free pass" to your partner, with the promise that they will make up by putting more effort in a later assignment.

3. **You are expected to prepare this assignment in LaTeX**.

4. If there is any problem in the team (such as lack of contribution, etc.), you must contact me as soon as the problem appears.

PROBLEM 1

Consider a system that associates satellites to their corresponding country of origin. The requirements of the system are as follows:

1. Satellite names are unique, i.e. no two different countries may have satellites that share a name.

2. Each satellite is owned by exactly one country, e.g. $KEPLER\_14$ is owned only by Canada.

We introduce types $SATELLITE\_NAME$ and $COUNTRY\_NAME$. The model of the system is captured by variable `registry`, as shown below:

$$
\begin{aligned}
&registry = \\
&\quad \{ \\
&\qquad KEPLER\_14 \mapsto Canada, \\
&\qquad KEPLER\_19 \mapsto Canada, \\
&\qquad GHGSAT\_C7 \mapsto Canada, \\
&\qquad IRIDIUM\_174 \mapsto USA, \\
&\qquad IRIDIUM\_179 \mapsto USA, \\
&\qquad STARLINK\_30079 \mapsto USA, \\
&\qquad COSMOS\_2569 \mapsto Russia, \\
&\qquad COSMOS\_2567 \mapsto Russia, \\
&\qquad YAOGAN\_36 \mapsto China, \\
&\qquad TIANZHOU\_6 \mapsto China \\
&\quad \}
\end{aligned}
$$

1. (2 pts) Is *registry* a binary relation? Explain and express this formally.

   Answer:

   Variable *registry* is a binary relation because is a subset of the Cartesian product of types $SATELLITE\_NAME$ and $COUNTRY\_NAME$, i.e.

   $$map \subseteq SATELLITE\_NAME \times COUNTRY\_NAME$$

2. (2 pts) In the expression $registry \in (...)$, what would the RHS be?

   Answer:

   $registry \in P(SATELLITE\_NAME \times COUNTRY\_NAME)$

3. (2 pts) Is *registry* a function? Explain and if Yes, determine the type of the function.

   Answer:

   Variable *registry* is a partial function which is neither injective nor surjective.

4. (2 pts) What is the value of the following expression:

   $$\{GHGSAT\_C7, IRIDIUM\_179, COSMOS\_2567\} \lhd registry$$

   Answer:

   Domain restriction selects pairs based on the first element. The value of the expression is as follows:

   $$\begin{Bmatrix} GHGSAT\_C7 \mapsto Canada, \\ IRIDIUM\_179 \mapsto USA, \\ COSMOS\_2567 \mapsto Russia \end{Bmatrix}$$

5. (2 pts) What is the value of the following expression:

$$registry \rhd \{Canada\}$$

Answer:

Range restriction selects pairs based on the second element. The value of the expression is as follows:

$$\begin{{}
KEPLER_14 \mapsto Canada,\\
KEPLER_19 \mapsto Canada,\\
GHGSAT_C7 \mapsto Canada
\end{}}$$

6. (2 pts) What is the value of the following expression:

$$\{IRIDIUM\_179, COSMOS\_2567, YAOGAN\_36\} \lhd registry$$

Answer:

Domain subtraction removes all elements from the domain of the relation. The value of the expression is as follows:

$$\begin{{}
KEPLER_14 \mapsto Canada,\\
KEPLER_19 \mapsto Canada,\\
GHGSAT_C7 \mapsto Canada,\\
IRIDIUM_174 \mapsto USA,\\
STARLINK_30079 \mapsto USA,\\
COSMOS_2569 \mapsto Russia,\\
TIANZHOU_6 \mapsto China
\end{}}$$

7. (2 pts) What is the value of the following expression:

$$registry \rhd \{USA, China\}$$

Answer:

Range subtraction removes all elements from the range of the relation. The value of the expression is as follows:

$$
\begin{Bmatrix}
KEPLER\_14 \mapsto Canada, \\
KEPLER\_19 \mapsto Canada, \\
GHGSAT\_C7 \mapsto Canada, \\
COSMOS\_2569 \mapsto Russia, \\
COSMOS\_2567 \mapsto Russia
\end{Bmatrix}
$$

8. (2 pts) What is the value of the following expression:

$$registry \oplus \{KEPLER\_19 \mapsto USA, IRIDIUM\_174 \mapsto Canada\}$$

Answer:

Relational overriding will produce a new binary relation:

$$
\begin{Bmatrix}
KEPLER\_14 \mapsto Canada, \\
KEPLER\_19 \mapsto USA, \\
GHGSAT\_C7 \mapsto Canada, \\
IRIDIUM\_174 \mapsto Canada, \\
IRIDIUM\_179 \mapsto USA, \\
STARLINK\_30079 \mapsto USA, \\
COSMOS\_2569 \mapsto Russia, \\
COSMOS\_2567 \mapsto Russia, \\
YAOGAN\_36 \mapsto China, \\
TIANZHOU\_6 \mapsto China
\end{Bmatrix}
$$

9. (6 pts) Assume that we need to add a new entry into the database table represented by *registry*. We have decided not to deploy a precondition. What could be the consequences to the system if we deployed a) **set union** and b) **relational overriding**?

Answer:

Adding with set union: In the case of set union, if the `(satellite, country)` pair already exists, then there will be no damage. If, however, there is another entry for the same satellite with a different country, then we will end up with a table with two entries for the same satellite. This has two consequences, both of which are violations of requirements: 1) "satellite must be owned by only one country", and variable *registry* being a function.

Adding with relational overriding: In the case of relational overriding, if the `(satellite, country)` pair already exists, then there will be no damage. If, however, there is another entry for the same satellite with a different country, then the record is replaced. This violates the requirements of the operation which is to add a new record.

PROBLEM 2

Consider a global package delivery service (such as FederalExpress, or UPS). Each package has a unique id called *Tracking_Number* : *TRACKING_TYPE* which is an alphanumerical string. The company maintains offices at various global locations, one of which is assigned as the place of origin for a package, upon registration. Another office is assigned as the final destination of the package. As the package travels to reach its final destination, it will arrive at possibly several intermediate office locations. At any time, one should be able to track the package to its most recent (i.e. current) location.

Let us introduce the following types:
$TRACKING\_TYPE$,
$LOCATION\_TYPE$, and
$PATH\_TYPE = LOCATION\_TYPE \times LOCATION\_TYPE$,
where a variable of type $PATH\_TYPE$ holds an origin-destination pair.

We also introduce the following variables:

1. *packages*: This variable holds all packages.
2. *description*: This variable holds, for each package, an association between its tracking number and its origin-destination information.
3. *history*: This variable holds, for each package, an association between its tracking number and all its locations, starting from its origin to the current one.
4. *track*: This variable holds, for each package, an association between its tracking number and its current location.

1. (3 pts) Provide a declaration of variable *packages* and explain your reasoning (both on the kind of the variable as well as on the appropriate type).

   Answer: $packages \ : \ \mathbb{P} \ TRACKING\_TYPE$.

   This is a variable that holds a **collection** of elements of type $TRACKING\_TYPE$.

2. (3 pts) What kind of variable is *description*? Provide a formal definition together with any and all applicable properties.

   Answer:

   (a) Variable *description* is a **collection variable** and it holds a binary relation. More specifically, the variable holds a subset of $TRACKING\_TYPE \times PATH\_TYPE$, i.e. it associates tracking information to origin-destination pairs.

   (b) Variable *description* is also a function. It is, in fact, a **partial function** as each element of some subset of the domain ($TRACKING\_TYPE$) maps to exactly one element (origin-destination pair) in the codomain ($PATH\_TYPE$), but it is neither injective nor surjective.

   (c) The function is not injective as there may be possibly several different packages that share the same path. It is not surjective as it is not necessary that every $path \in PATH\_TYPE$ would be associated with elements in $TRACKING\_TYPE$. Finally, by definition it is not bijective.

3. (3 pts) Describe what *data structure* you would deploy to model variable *description*. Note that you may not use a Dictionary. Should this be an ordered or an unordered structure? Discuss.

   Answer:

   Variable *description* is a set of pairs, where for each pair the first coordinate is a variable of type $TRACKING\_TYPE$, and the second coordinate is a variable of type $PATH\_TYPE$, i.e. $TRACKING\_TYPE, LOCATION\_TYPE \times LOCATION\_TYPE$.

4. (1 pt) Define variable *description* in Common LISP and populate it with some sample data.

5. (3 pts) Describe how you would validate variable *description*.

   Answer:

   We will need to iterate over the entire collection and make sure each variable that appears as the first coordinate in a pair, does not appear anywhere else, validate its property as a function.

6. (5 pts) Define a predicate function, `isfunctionp`, in Common LISP that reads a variable like *description* and indicates if the variable corresponds to a function or not.

7. (5 pts) Define a function in Common LISP that adds a new *tracking number* and *path* to *description*. You may need to define auxiliary functions in order to ensure the precondition.

8. (7 pts) What kind of variable is *track*? Provide a formal definition together with any and all applicable properties.

   Answer:

   (a) Variable *track* is a **collection variable** and it holds a binary relation. More specifically, the variable holds a subset of $TRACKING\_TYPE \times LOCATION\_TYPE$, i.e. it holds tracking-location pairs.

   (b) Variable *track* is also a function. It is, in fact, a **partial function** as each element of some subset of the domain ($TRACKING\_TYPE$) maps to exactly one element in the codomain ($LOCATION\_TYPE$), but it is neither injective nor surjective.

   (c) The function is not injective as there may be possibly several different packages that share the same current location. It is not surjective as it is not necessary that every $location \in LOCATION\_TYPE$ would be associated with elements in $TRACKING\_TYPE$. Finally, by definition it is not bijective.

9. (6 pts) Provide a formal specification of the state of the system in terms of a **Z specification schema**.

Answer: The state schema is shown below:

PackageDelivery

$$
\begin{array}{l}
packages : \mathbb{P}\ TRACKING\_TYPE \\
description : TRACKING\_TYPE \pfun PATH\_TYPE \\
history : TRACKING\_TYPE \pfun Seq(LOCATION\_TYPE) \\
track : TRACKING\_TYPE \pfun LOCATION\_TYPE \\
\hline
packages = \mathrm{dom}\ track
\end{array}
$$

10. (6 pts) Provide a schema for operation `RegisterPackagetOK` that adds a new package to the system. With the aid of success and error schema(s), provide a definition for operation `RegisterPackage` that the system will place in its exposed interface.

Answer: The schemas and operation definition are shown below:

RegisterPackageOK

$$
\begin{array}{l}
\Delta PackageDelivery \\
tracking\_number? : TRACKING\_TYPE \\
path? : PATH\_TYPE \\
\hline
tracking\_number? \notin packages \\
packages' = packages \cup \{tracking\_number?\} \\
description' = description \cup (tracking\_number? \mapsto path?) \\
track' = track \cup \{tracking\_number? \mapsto \pi_1(path?)\} \\
history(id)' = concat(history(tracking\_number?), list(\pi_1(path?)))
\end{array}
$$

NOTE: $\pi_{1,2}$ refers to the first or second coordinate in an ordered pair. We may also use generic functions such as *first_coordinate* and *second_coordinate*.

Success

$$
\begin{array}{l}
\Xi PackageDelivery \\
response! : MESSAGE \\
\hline
response! = 'Success'
\end{array}
$$

PackageAlreadyRegistered

$\Xi PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$response! : Message$

$tracking\_number? \in packages$
$response! = 'Error : Package\ already\ registered'$

$$RegisterPackage \mathrel{\hat{=}} (RegisterPackageOK \land Success) \oplus PackageAlreadyRegistered$$

11. (6 pts) Provide two alternative schemas for operation `UpdateLocationOK` that updates the current location of a given package. With the aid of success and error schema(s), provide a definition for operation `UpdateLocation` that the system will place in its exposed interface.

    Answer: The schemas and operation definition are shown below:

    Version 1: If we use a precondition ($current\_location? \neq track(tracking\_number?)$) to ensure that the location is indeed new, then we can add tracking information using set union:

UpdateLocationOK

$\Delta PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$current\_location? : LOCATION\_TYPE$

$tracking\_number? \in packages$
$current\_location? \neq track(tracking\_number?)$
$track' = track \cup \{tracking\_number? \mapsto current\_location?\}$
$history(id)' = concat(history(tracking\_number?), list(current\_location?))$

Version 2: Relational override makes precondition superfluous:

UpdateLocationOK

$\Delta PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$current\_location? : LOCATION\_TYPE$

$tracking\_number? \in packages$
$track' = track \oplus \{tracking\_number? \mapsto current\_location?\}$
$history(id)' = concat(history(tracking\_number?), list(current\_location?))$

PackageUnknown

$\Xi PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$response! : Message$

$tracking\_number? \notin packages$
$response! = 'Error : Package\ not\ registered'$

NotNewLocation

$\Xi PackageDelivery$
$current\_location? : LOCATION\_TYPE$
$response! : Message$

$current\_location? = track(tracking\_number?)$
$response! = 'Error : Not\ new\ location'$

$$UpdateLocation \mathrel{\hat{=}} (UpdateLocationOK \land Success) \oplus (PackageUnknown \lor NotNewLocation)$$

12. (6 pts) Provide a schema for operation `GetCurrentLocationOK` that returns the current location of a given package. With the aid of success and error schema(s), provide a definition for operation `GetCurrentLocation` that the system will place in its exposed interface.

Answer: The schemas and operation definition are shown below:

GetCurrentLocationOK

$$\Xi PackageDelivery$$
$$tracking\_number? : TRACKING\_TYPE$$
$$current\_location! : LOCATION\_TYPE$$

$$tracking\_number? \in packages$$
$$current\_location! = track(tracking\_number?)$$

$$GetCurrentLocation \mathrel{\widehat=} (GetCurrentLocationOK \land Success) \oplus PackageUnknown$$

13. (6 pts) Provide a schema for operation `GetAllPackages` that returns the tracking numbers of all packages at a given current location. With the aid of success and error schema(s), provide a definition for operation `GetAllPackages` that the system will place in its exposed interface.

Answer: The schemas and operation definition are shown below:

GetAllPackagesOK

$$\Xi PackageDelivery$$
$$location? : LOCATION\_TYPE$$
$$list! : \mathbb{P}\ TRACKING\_TYPE$$

$$location? \in \operatorname{ran} track$$
$$list! = \operatorname{dom}(track \rhd \{location?\})$$

LocationUnknown

$\Xi PackageDelivery$
$location? : LOCATION\_TYPE$
$response! : Message$

$location? \notin \operatorname{ran} track$
$response! = \text{'}Error : Location\ unknown\text{'}$

$$GetAllPackages \mathrel{\hat{=}} (GetAllPackagesOK \land Success) \oplus LocationUnknown$$

14. (6 pts) Provide a schema for operation `IsDeliveredOK` that returns true or false depending on whether or not the package has been delivered to its destination office. With the aid of success and error schema(s), provide a definition for operation `IsDelivered` that the system will place in its exposed interface.

    Answer: The schemas and operation definition are shown below:

IsDeliveredOK

$\Xi PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$current\_location! : LOCATION\_TYPE$

$tracking\_number? \in packages$
$track(tracking\_number) = \pi_2(description(tracking\_number?))$

15. (6 pts) Provide a schema for operation `GetHistory` that returns a history of a package given its tracking number. With the aid of success and error schema(s), provide a definition for operation `GetHistory` that the system will place in its exposed interface.

    Answer: The schemas and operation definition are shown below:

GetHistoryOK

$\Xi PackageDelivery$
$tracking\_number? : TRACKING\_TYPE$
$package\_history! : Seq(LOCATION\_TYPE)$

$tracking\_number? \in packages$
$package\_history! = history(tracking\_number?)$

$$GetHistory \mathrel{\hat{=}} (GetHistoryOK \land Success) \oplus PackageUnknown$$

16. (6 pts) Provide a schema for operation `SignOffOK` that deletes a package record upon delivery. With the aid of success and error schemas, provide a definition for operation `SignOff` that the system will place in its exposed interface.

    Answer: The schemas and operation definition are shown below:

$$\begin{array}{|l}
\hline
SignOffOK \\
\Delta PackageDelivery \\
tracking_number? : TRACKING_TYPE \\
\hline
tracking_number? \in packages \\
track' = \{tracking_number?\} \lhd track \\
description' = \{tracking_number?\} \lhd description \\
history' = \{tracking_number?\} \lhd history \\
\hline
\end{array}$$

$$SignOff \mathrel{\hat{=}} (SignOffOK \land Success) \oplus PackageUnknown$$

4 What to submit

You must prepare all your solutions in LaTeX. This will be your main submission file. Produce a `pdf` file of your main submission. All Common LISP functions are supporting documents. Place all files (`.tex`, `.pdf`, `.lisp`) into a folder that is named after you and your partner, where the name of the person to submit goes first, e.g. if Roger Waters and David Gilmour were partners and Roger will be the one to submit, then the folder is called `Waters-Gilmour`. Zip your folder and submit it at the Electronic Assignment Submission portal at

(`https://fis.encs.concordia.ca/eas`)

under **Assignment 2**.

END OF ASSIGNMENT.