Z Notations

Dr. Rong Qu

rxq@cs.nott.ac.uk

http://www.cs.nott.ac.uk/∼rxq/#g53fsp

Introduction

We use mathematical notation so that we will be able to

prove certain properties of the system directly from the specification. i.e. it is consistent and it is complete

answer questions about the system. i.e. "Can such and such a situation ever arise?"

produce computer programs directly from the specification, or confirm that an existing program conforms the specification

But

There remains always, of course, the problem of proving that our mathematics actually represents the real-world problem that we are trying to represent

Schema

The specification is broken down into small units called schema

Each schema will have

declaration part and

a logical or predicate part

Identifiers

Identifiers followed by a prime ' indicate the values of objects after the action has taken place

Identifiers followed by a question mark ? indicate input values identifiers

Identifiers followed by a exclamation ! indicate output values

A State Schema

Assume a particular possible state of our system to be

$$known = \{Joy, Eric\}$$

known as a set of names

A State Schema

$$height = \{(Joy, 6feetand3inches),\\ (Eric, 5feetand2inches)\}$$

a function mapping names to heights

$$weight = \{(Joy, 7stonesand2pounds),\\ (Eric, 17stonesand10pounds)\}$$

a function mapping names to weights

A State Schema

The function $height$, $weight$ could be equally written

$$height = \{(Joy \mapsto 6feetand3inches),$$
$$(Eric \mapsto 5feetand2inches)\}$$

$$weight = \{(Joy \mapsto 7stonesand2pounds),$$
$$(Eric \mapsto 17stonesand10pounds)\}$$

State-space Schema

describes the logic of the overall state of our system

$$[\mathrm{NAME}, \mathrm{HEIGHT}, \mathrm{WEIGHT}]$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{Height\_and\_Weight}} \\
known\_height : \mathbb{P}\ NAME \\
known\_weight : \mathbb{P}\ NAME \\
height : NAME \rightarrow\!\!\!\!\!+\ HEIGHT \\
weight : NAME \rightarrow\!\!\!\!\!+\ WEIGHT \\
\hline
known\_height = \mathrm{dom}\ height \\
known\_weight = \mathrm{dom}\ weight \\
\hline
\end{array}
$$

The Declaration Part

The initial line

$$[NAME, HEIGHT, WEIGHT]$$

declares that $NAME, HEIGHT$ and $WEIGHT$ are three basic data types

we will not be defining them further in this specification

The Declaration Part

$$known\_height : P\ NAME$$
$$known\_weight : P\ NAME$$
$$height : NAME \rightarrow\!\!\!\!\!+ \ HEIGHT$$
$$weight : NAME \rightarrow\!\!\!\!\!+ \ WEIGHT$$

declares that

$known\_height$ and $known\_weight$ are to be sets of $NAMEs$

$height$ and $weight$ are to be partial functions which will act on a $NAMEs$ to give a $HEIGHT$ or a $WEIGHT$ respectively

The Predicate Part

The lower part of the schema

$$known\_height = dom\ height$$
$$known\_weight = dom\ weight$$

consists of logical statements which define the system

the set $known\_height$ is to be exactly equal to the domain of the $height$ function

the set $known\_weight$ is to be exactly equal to the domain of the $weight$ function

The Predicate Part

This part of the schema declares logical statements which are always true, and are $\mathrm{invariants}$ of the system

If there are several statements in the predicate part, their order is immaterial; they all represent conditions which must be true

Note that $known\_height$ and $known\_weight$ are derived objects

Operations and Their Schema

We can declare the action of adding a new height to the list as the schema

This is known as an operation schema since it describes the change to the system brought about by a given operation or event

Operations and Their Schema

$$
\begin{array}{|l}
\hline
\textit{New\_Height} \\
\Delta Height\_and\_Weight \\
name? : NAME \\
hgt? : HEIGHT \\
\hline
name? \notin known\_height \\
height' = height \cup \{name? \mapsto hgt?\} \\
weight' = weight \\
\hline
\end{array}
$$

Included Schema

$\Delta Height\_and\_Weight$

state that the schema $Height\_and\_Weight$ will be used with both its declarations and predicates.

The symbol Δ in front of the name indicates that we wish to use this schema in association with a state change.

In any state change, a primed identifier indicates the value after change, the unprimed identifier represents the value before the change.

The Δ Inclusion

By including the schema using

$$\Delta Height\_and\_Weight$$

we are automatically including all the declarations

$$known\_height, known\_height' : P\ NAME$$
$$known\_weight, known\_weight' : P\ NAME$$
$$height, height' : NAME \pfun HEIGHT$$
$$weight, weight' : NAME \pfun WEIGHT$$

The Δ Inclusion

The $\Delta Height\_and\_Weight$ also causes the predicates from $Height\_and\_Weight$ to be included in the predicate part

$$known\_height = dom\ height$$
$$known\_weight = dom\ weight$$
$$known\_height' = dom\ height'$$
$$known\_weight' = dom\ weight'$$

New_Height

We also declare that there will be an input argument $name?$ of type $NAME$, and a second input argument $hgt?$ of the type $HEIGHT$.

Pre- and Post- Conditions

$$name? \notin known\_height$$

This predicate is known for obvious reasons as a pre-condition, defining conditions which must hold when the operation starts

The second and third predicates are post-conditions

$$height' = height \cup \{name? \mapsto hgt?\}$$
$$weight' = weight$$

Pre- and Post- Conditions

We could also describe

$$known\_height = dom\ height$$
$$known\_weight = dom\ weight$$

as pre-conditions, and

$$known\_height' = dom\ height'$$
$$known\_weight' = dom\ weight'$$

as post-conditions

Consistency Checks

After the operation has taken place, we would expect that

$$known\_height' = \\ known\_height \cup \{name?\}$$

Observation Schema

An observation schema is one which provides information about the state of the system, without changing the state

To find a given person's weight, for example, we use the schema

Observation Schema

$$
\begin{array}{|l}
\hline
\textit{Find\_Weight} \\
\Xi Height\_and\_Weight \\
name? : NAME \\
wgt! : WEIGHT \\
\hline
name? \in known\_weight \\
wgt! = weight\ name? \\
\hline
\end{array}
$$

Invariant Ξ Inclusion

$\Xi Height\_and\_Weight$

is an extension of the $\Delta Height\_and\_Weight$ idea introduced earlier

It introduces the $\Delta Height\_and\_Weight$ schema and, since we have an observation schema with no change in the system data, it provides the additional predicates.

Invariant Ξ Inclusion

$$known\_height' = known\_height$$
$$known\_weight' = known\_weight$$
$$height' = height$$
$$weight' = weight$$

The exclamation mark in $wgt!$ indicates that this is an output object.

A Query Schema

It is possible to construct a schema which will have as inputs as specific height and weight and will have as output the set of people who have both that height and that weight

A Query Schema

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Who\_is\_that\_high\_and\_that\_tall \quad\quad\quad\quad\quad\quad\quad\quad}} \\
\Xi Height\_and\_Weight \\
hgt? : HEIGHT \\
wgt? : WEIGHT \\
names! : \mathbb{P}\ NAME \\
\hline
names! = \{n : known\_height \mid height\ n = hgt?\} \\
\qquad \cap \{n : known\_weight \mid weight\ n = wgt?\} \\
\hline
\end{array}
$$

Error Messages and the Like

We need a free type definition as follows

$$
\begin{aligned}
REPORT ::= {} & ok \mid height\_already\_anown \mid \\
& height\_not\_known \mid weight\_already\_known \mid \\
& weight\_not\_know
\end{aligned}
$$

We need one extra schema to define a successful result

$$
\begin{array}{|l}
\hline
\textit{Success} \\
\hline
report! : REPORT \\
\hline
report! = ok \\
\hline
\end{array}
$$

Error Messages and the Like

$New\_Height \wedge Success$

gives a schema which adds

$report! : REPORT$

to the New_Height predicate part.

$report! = ok$

to the New_Height declaration part.

$Height\_Already\_Known$ **Schema**

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Height\_Already\_Known}} \\
\hline
\Xi Height\_and\_Weight \\
name? : NAME \\
report! : REPORT \\
\hline
name? \in known\_height \\
report! = height\_already\_known \\
\hline
\end{array}
$$

$Height\_Already\_Known$ Schema

Now combine the schema

$$(New\_Height \wedge Sccuess) \vee Height\_Already\_Known$$

A full definition is

$$\begin{aligned}
&Full_New_Height \mathrel{\widehat=} \\
&\qquad (New_Height \wedge Sccuess) \vee \\
&\qquad Height_Already_Known
\end{aligned}$$

The Full Equivalent

$$
\begin{array}{l}
\textit{Full\_New\_Height} \\
\hline
known\_height, known\_height' : \mathbb{P}\ NAME \\
known\_weight, known\_weight' : \mathbb{P}\ NAME \\
height, height' : NAME \pfun HEIGHT \\
weight, weight' : NAME \pfun WEIGHT \\
name? : NAME \\
hgt? : HEIGHT \\
report! : REPORT \\
\hline
(name? \notin known\_height \land height' = height \cup \{name? \mapsto hgt?\} \land weight' = weight \land \\
\quad known\_height = \mathrm{dom}\ height \land known\_weight = \mathrm{dom}\ weight \land \\
\qquad known\_height' = \mathrm{dom}\ height' \land \\
\qquad\quad know\_weight' = \mathrm{dom}\ weight' \land report! = ok) \\
\lor\ (name? \in known\_height \land height' = height \land weight' = weight \land \\
\quad known\_height = \mathrm{dom}\ height \land known\_weight = \mathrm{dom}\ weight \land \\
\qquad known\_height' = \mathrm{dom}\ height' \land know\_weight' = \mathrm{dom}\ weight' \land \\
\qquad\quad report! = heigh\_already\_known)
\end{array}
$$

$Weight\_Not\_Known$ Schema

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Weight\_Not\_Known}} \\
\hline
\Xi Height\_and\_Weight \\
name? : NAME \\
report! : REPORT \\
\hline
name? \notin known\_weight \\
report! = weight\_not\_known \\
\hline
\end{array}
$$

$Full\_Find\_Weight$ **Schema**

For a full version of the $Find\_Weight$ schema, we can define

$$Full\_Find\_Weight \mathrel{\widehat=} \\ (Find\_Weight \land Success) \lor Weight\_Not\_Known$$

Pre- and Post- Conditions

The transaction operation will update the value of the global variable $till\_state$ upon input of one integer parameter $transaction$.

If $transaction$ is greater than or equal to 1000, then $till\_state$ is to be set to 2; otherwise $till\_state$ will be set to the value 1.

The value of $transaction$ will be greater than zero on entry, and will not be changed by the procedure. The value of $till\_state$ on entry will be 1 or 2.

Pre- and Post- Conditions

Pre-condition

$$transaction \geq 0 \land (till\_state = 1 \lor till\_state = 2)$$

Post-condition

$$\begin{aligned}&(transaction \geq 1000 \Rightarrow till\_state' = 2)\\&\quad\land (transaction < 1000 \Rightarrow till\_state' = 1)\\&\quad\land transaction' = transaction)\end{aligned}$$

Notational Difference

$$\forall\, x(is\_an\_integer(x) \Rightarrow Pred(x))$$

The statement "For all values of x in the set S the logical expression $P(x) \wedge Q(x)$ holds" is written

$$\forall\, x : S \bullet P(x) \wedge Q(x)$$

Notational Difference

For sets consisting of all the integers in a given numeric range, we write the set of integers from 1 to 100 inclusive as "$1..100$".

For the set of natural numbers including zero (0, 1, 2, ...) we write N

For the natrual numbers starting at 1 we write N_1

For all integers (nagitive, zero and positive) Z

Notational Difference

We also have multiple variables ranging over the same set

$$\forall\, i, j, k : S_1 \bullet ...$$

$$\forall\, i, j, k : S_1;\, x, y, z : S_2 \bullet ...$$

Unique Exist Quantifiers

Unique exists

$$\exists ! x : S \bullet < logical\ \ expr >$$

There exists exactly one x in S such that ...

$$\forall\, n : N \bullet \exists ! m : N \bullet m = succ(n)$$

Every natual number has a unique number which follows it.

Counting Quantifier in Z

How many exist. This is written

$$\Omega\ x : S \bullet < logical\ \ expr >$$

$$\exists\, x : S \bullet P(x) \Leftrightarrow (\Omega x : S \bullet P(x)) > 0$$

$$\exists! x : S \bullet P(x) \Leftrightarrow (\Omega x : S \bullet P(x)) = 1$$

$$\Omega\ account : all\_accounts \bullet balance\ \ account < 0$$

Summation Quantifier

Summation

$$\sum x : S \bullet < numeric\ expr >$$

$$\sum account : all\_accounts \bullet balance\ account$$

Note 1

You should be careful using the above that you use logical and numerical expressions.

$\forall$, $\exists$ and Ω are followed by a logical expression

$\sum$ uses a numeric expression

and results

$\forall$ and $\exists$ deliver logical results

Ω and $\sum$ deliver numeric results

in their correct places

Note 2

Conventions for the empty set (written $\{\}$) are that

$\forall\, x : \{\} \bullet P(x)$ is true

$\exists\, x : \{\} \bullet P(x)$ is false

$\exists! x : \{\} \bullet P(x)$ is false

$\Omega x : \{\} \bullet P(x)$ is zero

$\sum x : \{\} \bullet N(x)$ is zero

Summary

Schema Introduction

State Schema (Declaration & predicate parts)

Operation Schema (inclusion Δ)

Observation (invariant inclusion Ξ)

Others

Error message, Pre- and post- condition, Notational Differences