Modeling Class Diagrams in Z

MohammadReza Mousavi, Michel Reniers

In Section 5.6 of the reader, we introduced a slight extension of Z notation for capturing class diagrams. Note that although we are inspired by the notation of Object-Z, we do not precisely follow this notation. This note provides some additional information regarding our notation for modeling class diagrams in Z.

1 Classes and Objects

As far as a single class is concerned the notation is straight-forward: the attributes of the class are captured in the state schema (an unnamed schema placed within the class definition) and for each method, an operation schema is defined. An "Init" schema defines the constructor of the class.

When it comes to using the definition of classes and methods in defining other classes, the situation might be less clear. If we quote the name of one class in another class definition, we get an exact copy of the former in the latter. This results in taking the union of the declarations and conjunctions of predicates in all schemata with identical names.

For example, consider the following specification for class *A*:

A

$$a : \mathbb{Z}$$

$$a > 0$$

Init

$$a' \leq 5$$

IncA

$$\Delta(a)$$

$$a' = a + 1$$
$$a' \leq 5$$

Class *B*, defined below, includes the definition of *A*:

B

A

$$b : \mathbb{Z}$$

$$b < 5$$

Init

$$b' = 0$$

By definition, class *B* is equivalent to the following specification:

B

$a : \mathbb{Z}$
$b : \mathbb{Z}$

$a > 0$
$b < 5$

INIT

$a' \leq 5$
$b' = 0$

IncA

$\Delta(a)$

$a' = a + 1$
$a' \leq 5$

Similarly, when we refer to operation schema *Op* of class *Cl* by writing *Cl.Op*, we get an exact copy of its definition. Thus, the operation schema *A.IncA* has the following specification.

A.IncA

$\Delta(a)$

$a' = a + 1$
$a' \leq 5$

Classes can be used as a type and thus variables (objects) of a class type can be instantiated. For example, one can write:

$o : A$

For an attribute a, $o.a$ is considered as the value of a in the instance o. For an operation schema *op*, $o.op$ refers to a copy of the operation schema *op* in *C* in which all before- and after-state variables are prefixed with $o.$, thus, for example, if o is instance of class *A*, *o.IncA* is by definition:

o.IncA

$\Delta(o.a)$

$o.a' = o.a + 1$
$o.a' \leq 5$

2 Re-Using Operation Schemata

Re-use is one of the corner-stones of object-oriented design. Thus, we need to re-use methods from classes in the definition of other classes. This is achieved by using the notations *Class.Op* or *object.Op* and then linking the state of the class to the before- and after-state specified by the schema *Op*. Suppose that class *C* has an instance o of class *A* and we would like to "call" the schema *IncA* to set the after-state value of o'. This is achieved as follows.

C

$o : A$

IncC

$\Delta(o)$

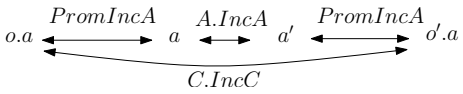
$\exists a, a' : \mathbb{Z} \bullet O.IncA \land PromIncA$

PromIncA

$\Delta(o)$
$a' : \mathbb{Z}$

$o.a = a$
$o'.a = a'$

Note that the schema *A.IncA* defines the relation between a and a', while in *C.IncC* we are interested in the relation between $o.a$ and $o'.a$; thus, we used a schema called *PromIncA*, which links $o.a$ and a on one hand, and $o'.a$ and a', on the other hand. The relationship established by different schemata is captured in the following figure.



The following calculations show why *C.IncC* indeed defines the right after-state for *C*.

IncC

$\Delta(o)$

$\exists a, a' : \mathbb{Z} \bullet O.IncA \land PromIncA$

$=$ Definitions of *IncA* and *PromIncA*

IncC

$\Delta(o)$

$\exists a, a' : \mathbb{Z} \bullet$
$a' = a + 1$
$a' \leq 5$
$o.a = a \land$
$o'.a = a'$

$= 2 \times$ One point rule

IncC

$\Delta(o)$

$\exists a, a' : \mathbb{Z} \bullet$
$o'.a = o.a + 1$
$o'.a \leq 5$