5 SCHEMAS

In the last chapter we saw how the values of variables may be constrained by predicates and how satisfying a predicate defines a set. The types of predicate discussed included:

=, $\in$ equals and membership
<, > the relations less than and more than
$\wedge$, $\vee$ the connectives conjunction (and) and disjunction (or)

Now we see how to combine declarations and predicates into structures called schemas. A schema represents a system's state. A collection of schemas models the behaviour of a computer system. We look at pre-conditions.

5.1 SYSTEM

The system we shall look at is a simple counter. A counter would be used, for example, to count the number of vehicles passing a census point, the number of people entering a stadium or the number of fleas in a bird's nest. When the counter is clicked, its value is increased by one. When the reset button is pressed, the counter's value becomes zero.

5.2 STATE

The contents of a system's memory are called its state. For our counter system that would be the current value of the count together with the maximum it can reach.

We have the initial state when the count is zero, maximum is 9999 (say).

Initial state: *count* = 0, *maximum* = 9999

We have an interim state when the count is between zero and the maximum value that count can have.

Interim state: *count* = 147, *maximum* = 9999

We have an end state when the count has reached it maximum value and cannot be advanced any further.

End state: *count* = 9999, *maximum* = 9999

Effectively, the state of a system is the collection of values stored in its variables.

EXERCISE 5.1

A cash point machine has a number of £10 notes available for issue to customers. Describe three distinct states of the cash point machine.

5.3 ABBREVIATION DEFINITION

Count and maximum are whole numbers. We could represent them as integers of type $\mathbb{Z}$, but that will allow negative values for count and maximum. A solution would be to introduce $\mathbb{N}$ (say fat N) as an abbreviation for the set of all natural (integer) counting numbers including zero but excluding negative values.

$$\mathbb{N} == \{ n : \mathbb{Z} \mid n \geqslant 0 \}$$

Declaring $\mathbb{N}$ in this way allows us to use the relational operators < and > as well as =. The type of $\mathbb{N}$ is $\mathbb{Z}$.

If we had introduced $\mathbb{N}$ as a given set like this:

$$[\mathbb{N}]$$

and if we had variables $a, b : \mathbb{N}$, then for sure we could write $a = b$, but we could not write, for example, $a < b$. The only number type that is part of the Z Notation is $\mathbb{Z}$.

The == is known as the definition symbol. We cannot use = because it is reserved for the equals predicate.

EXERCISE 5.2

Introduce and define the set $\mathbb{N}_1$, the set of all natural counting numbers from 1 upwards, excluding zero and negative numbers.

5.4 AXIOMATIC DEFINITION

The maximum value that count can reach is set or fixed. We express this in an axiomatic definition or description.

$$maximum : \mathbb{N}$$
$$\hline$$
$$maximum = 9999$$

An axiomatic definition has two parts.

Above the dividing line we have a declaration - $maximum : \mathbb{N}$.

Below the dividing line we have a predicate - $maximum = 9999$.

Remembering the definition before use principle, an axiomatic definition is effective from the point where it is defined; you cannot use it before you have defined it.

The predicate is optional. We could have written

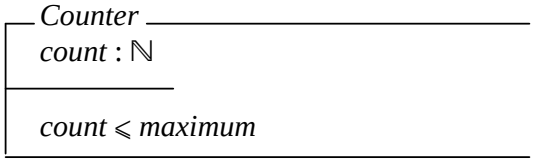
$$maximum : \mathbb{N}$$

and left its actual value unsaid.

5.5 SYSTEM STATE SCHEMA

Variables are derived from what an object has. A counter has just one variable that we shall name *count*. *count* can equal, but never exceed, *maximum*.

A *schema* is a set of variables together with a set of predicates constraining those variables. A schema is drawn as an open box - see below.



Our schema, named *Counter* and shown above, has one variable named *count* and one predicate *count* ⩽ *maximum*. *count* is constrained to be a value between zero (because its values are drawn from the set ℕ) and *maximum* (whatever value that is) inclusive.

We (nearly) always start schema names with an uppercase letter. A schema name cannot contain spaces.

The variables are declared above the line that separates declarations from predicates. The variables' scope is strictly confined to the schema. This means that you cannot refer *Counter*'s variable *count* outside the schema.

The predicates are described below the dividing line. If there were more than one predicate, they would all be joined by conjunction - the ∧ symbol is assumed - by default.

The variables of a schema are described as its *components*. The predicates of a schema are described as its *property*.

The schema shown is the system state schema. No matter what process we define, add 1 to *count*, report on the value stored in *count*, subtract one from *count*, for example, we are obliged to ensure that the predicate *count* ⩽ *maximum* is always satisfied.

The schema that describes a system's variables, and predicates on those variables, is called a *system state schema*. It represents an object's state, the collection of values stored in its variables at any moment in time. Its variables are called *state variables*.

The processes that we shall describe in other schemas both update and report the values stored in the state variable *count*.

5.6 THE INITIAL STATE

We describe the state the system is in when it is first started. The *InitCounter* schema defined below describes the initial state of *Counter*.

InitCounter

Counter

$count = 0$

The line *Counter* says include all the variables defined in the *Counter* schema (*count* in this instance) and all the predicates ($count \leqslant maximum$ in this instance). To illustrate the point we write out *InitCounte*r in full below.

InitCounter

$count : \mathbb{N}$

$count \leqslant maximum$

$count = 0$

The initial state is that *count* is zero.

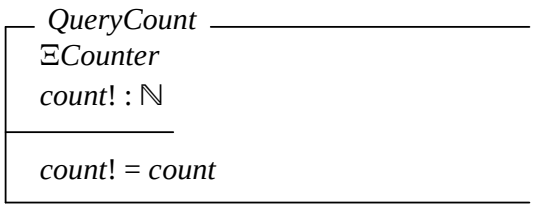
We can see at a glance that the predicates are consistent: *count* is zero and *count* is less than or equal to *maximum* - there are no contradictions.

The mechanism of schema inclusion, just by quoting the schema's name, allows us to

- concentrate on just a few things at a time
- keep schemas small and simple
- re-use pre-written schemas in different contexts or situations.

5.7 QUERY SYSTEM STATE

The schema *QueryCount* shown below outputs the current value of *count*.



Ξ*Counter* (say Xi Counter) says include all the variables and predicates defined in the *Counter* schema; the values stored in these variables will not change.

The declaration *count*! : ℕ says count output is drawn from the set ℕ. The ! mark stands for output.

The predicate *count*! = *count* says count output is the same as the (system variable) *count*.

5.8 CHANGE SYSTEM STATE

Not only does a schema define and report on a system's state, it also describes changes in that state.

The value of *count* does not remain the same forever. From time to time the counter will be clicked and the value stored in *count* will be moved on. The *Click* schema shown below updates the *count* system state variable.

Click

$\Delta Counter$

$count < maximum$

$count' = count + 1$

$\Delta Counter$ (say delta Counter) says:

- include all the variables and predicates defined in the *Counter* schema
- the values stored in some (or all) of these variables may well change
- decorate each variable with a ' to represent the state after an update has taken place

$count' = count + 1$ says the updated value of *count* is the same as the original value of *count* plus one. So, for example, if *count* was 4, *count'* would be 5. *count'* and *count* are both the same variable. The ' in *count'* means the state of *count* after an update has taken place.

The essential requirement is that $count \leqslant maximum$ at all times. If $count = maximum$ how can $count' = count + 1$ be met? We need ensure that *count* is strictly less than *maximum* before adding 1 to it. We do this by specifying the *pre-condition*

$$count < maximum$$

A pre-condition describes what must be true for a successful outcome. If a pre-condition is not met have no way of knowing what the outcome will be; it could be anything.

Consider *Cancel*, which undoes a *Click* operation. The essential operation here is:

$$count' = count - 1$$

But the least value *count* can have is zero. If *count* is already zero you cannot subtract one from it. So the precondition for a successful *Click* operation must be $count > 0$.

EXERCISE 5.4

1 Write and fully explain a schema named *Cancel*, which undoes a *Click* operation.

2 Write and fully explain a schema named *Reset*, which sets *count* to zero.

3 Write and fully explain a set of schemas for the car park system described below.

A car park has a capacity - a limited number of spaces. A count is maintained of the number of cars currently occupying spaces. When a car enters the car park, the count is increased. When a car leaves the car park the count is decreased. The system outputs the number of spaces left in the car park.

REVIEW

We combined declarations and predicates into structures called schemas. We saw that a schema represents a system's state. We saw how a collection of schemas models the behaviour of a computer system. We looked at pre-conditions. Next we look at set types. A set type variable represents a collection of similar objects.

BIBLIOGRAPHY

BARDEN R., STEPNEY S. & COOPER D 1994 Z *In Practice Prentice* Hall pp 370
JACKY J. 1997 *The Way of Z* Cambridge University Press pp 49, 122
SPIVEY J.M. 1992 *The Z Notation* Prentice Hall pp 7, 28, 48, 82, 128
WOODCOCK J. & DAVIES J. 1996 *Using Z: Specification, Refinement and Proof* Prentice Hall pp 203