Formal Methods

Z Language

- Z is a formal specification language based on Zermelo set theory. It was developed at the Programming Research Group at Oxford University in the early 1980s
- Z specifications are mathematical and employ a classical two-valued logic. The use of mathematics ensures precision and allows inconsistencies and gaps in the specification to be identified

Z Language

- Z is a “model-oriented” approach with an explicit model of the state of an abstract machine given, and operations are defined in terms of this state.
- Its mathematical notation is used for formal specification, and the schema calculus is used to structure the specifications. The schema calculus is visually striking, and consists essentially of boxes, with these boxes or schemas used to describe operations and states.
- The schemas may be used as building blocks and combined with other schemas..

Schema

- The schema calculus is a powerful means of decomposing a specification into smaller pieces or schemas.
- This helps to make Z specifications highly readable, as each individual schema is small in size and self-contained.

Schema

- A schema describes both the static and the dynamic aspects of a system.
- Static
  - Individual states
  - Invariants maintained

Schema

- Dynamic
  - State Transitions
  - Events (operations)
  - Transformations (relationship of input to output)

Operation

- Operations are defined in a precondition/postcondition style.
- A precondition must be true before the operation is executed, and the post condition must be true after the operation has executed.
- System invariant is the property that must be true at all times.

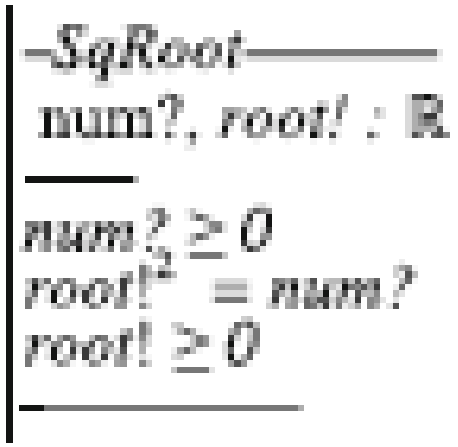
Z language

- Z is a typed language and whenever a variable is introduced its type must be given.
- A type is simply a collection of objects, and there are several standard types in Z.

Conventions

- Various conventions are employed within Z specification.
- for example
- v? indicates that v is an input variable;
- v! indicates that v is an output variable.
- The variable num? is an input variable and root! is an output variable for the square root example ahead.
- The notation in a schema indicates that the operation Op does not affect the state, whereas the notation Δ in the schema indicates that Op is an operation that affects the state.

Example



Example Explanation

- The precondition for the specification of the square root function above is that num?> 0; i.e. the function SqRoot may be applied to positive real numbers only.
- The postcondition for the square root function is root!2 = num? and root! 0. That is, the square root of a number is positive and its square gives the number.
- Postconditions employ a logical predicate which relates the prestate to the poststate, with the poststate of a variable being distinguished by priming the variable, e.g. v′.

Z Language

- The Z language also allows us to combine and relate separately defined schemas in a mathematically logical fashion.

Z Language

- To learn how to speak any language, we first need to learn some vocabulary and some simple rules of grammar. Both the vocabulary and the grammar of Z should be largely familiar to you.

Z Language

- Some of these symbols and concepts are unique to Z but most are in general use elsewhere in mathematics.

- In this treatment we use the ObjectZ dialect of Z, one that allows for the notion of class, inheritance and a few other OO concepts

Z Language

- X = = Y Y stands for X
- $\mathbb{Z}$ Set of Integers
- $\mathbb{N}$ Set of natural numbers (≥ 0)
- $\mathbb{N}1$ Set of positive integers (≥ 1)
- $t \in S$ t is an element of set S
- $t \notin S$ t is not an element of set S
- $S \subset T$ S is contained in T

Z Language

- $S \subseteq T$ S is strictly contained in T ($S \neq T$)
- $S \not\subset T$ S is not contained in T
- $S \cap T$ Set of intersection of S and T
- $S \cup T$ Set of the union of S and T
- $\mathbb{P}S$ Powerset of S: the set of all subsets of S
- $\mathbb{F}S$ Finite powerset of S: the set of all finite subsets of S
- $\varnothing$ Or { } The null or empty set

Z Language

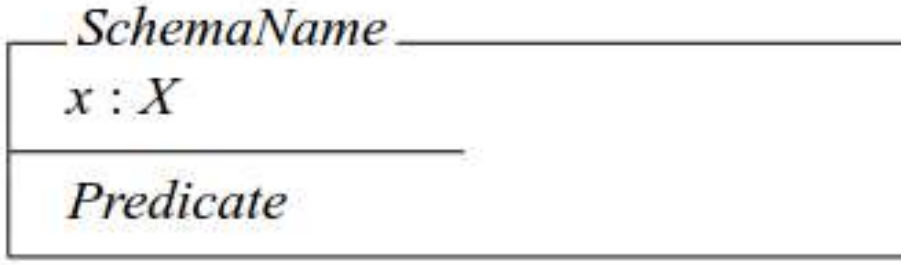
- S\T Difference: elements that are in S but not in T
- #S Size or cardinality: number of elements in S

Z-Schema Conventions

- Delta
  - Denoted by the Greek literal (Δ)
  - Used to extend the schema components to indicate update operations, i.e. changes in state variables (updating operations).
- "Xi"
  - Denoted by the Greek literal (Ξ)
  - Used to indicate that stored data is not affected, i.e. enquiry operations.

Z language - Schema

- A Z schema consists of a name, a declaration of variables, and a predicate



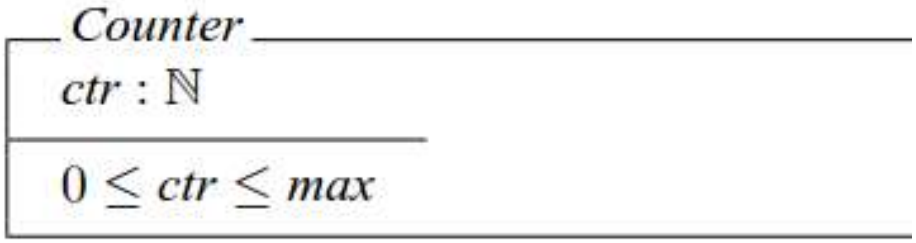
Z language - Schema

- A system specification in Z consists of some state variables, an initialisation, and a set of operations on the state variables.

- The state variables will also have some invariants associated with them representing “healthiness conditions” which must always be satisfied.

Z language - Schema

- For example, the state variables of a counter system may be specified using the following schema:

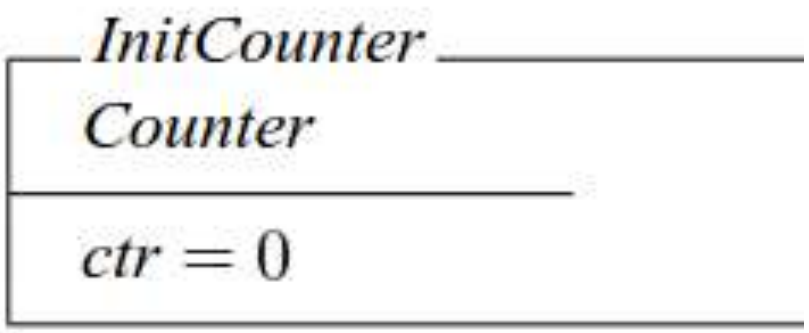


Z language - Schema

- Here, ctr is declared to be a natural number and the predicate part describes an invariant that must be satisfied by ctr, the state variable of the system

Initialization

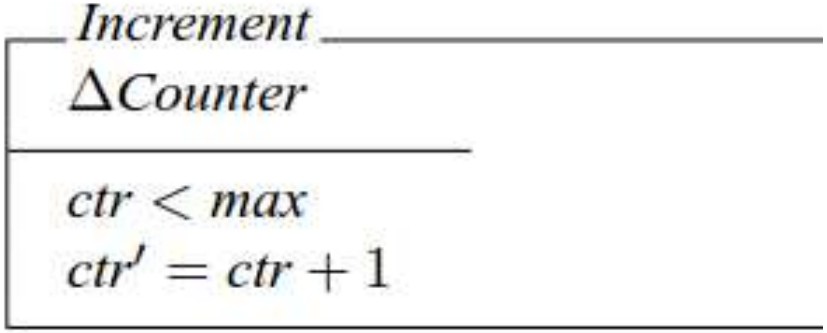
- An initialisation may be specified as follows:



Operation

- An operation is specified in Z with a predicate relating the state before and after the invocation of that operation.

- For example, an operation to increment the counter until some maximum value, may be specified as follows



Operation

- The declaration ΔCounter means that the state Counter is changed by the operation

- In the predicate, the new value of a variable is changed (ctr’), while the old value is unchanged (ctr).

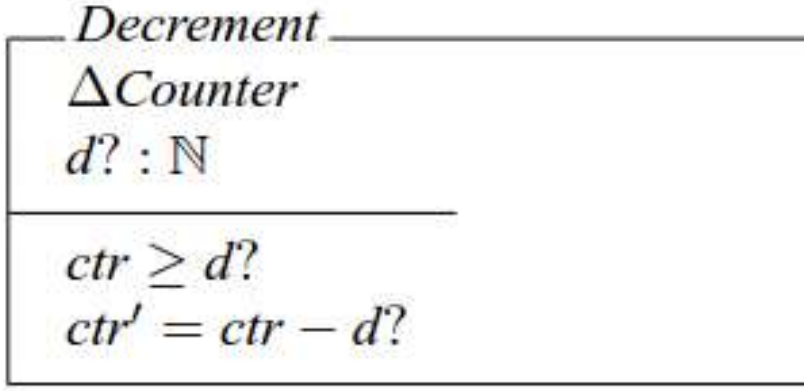
Operation

- So the predicate states that the new value of the counter, ctr', is the old value plus one.

- As well as changing the state variables, an operation may also have input and output parameters. Input parameter names are usually suffixed with '?', while output parameter names are suffixed with '!'.

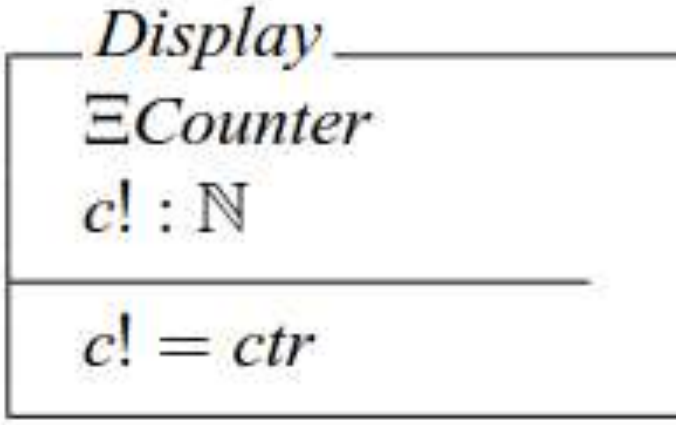
Example

- For example, the following operation for decrementing the counter has as an input parameter, the amount by which the counter should be decremented:



Example

- The following operation has an output parameter which is the value of the counter



Example Specification: Check-In/Check-Out

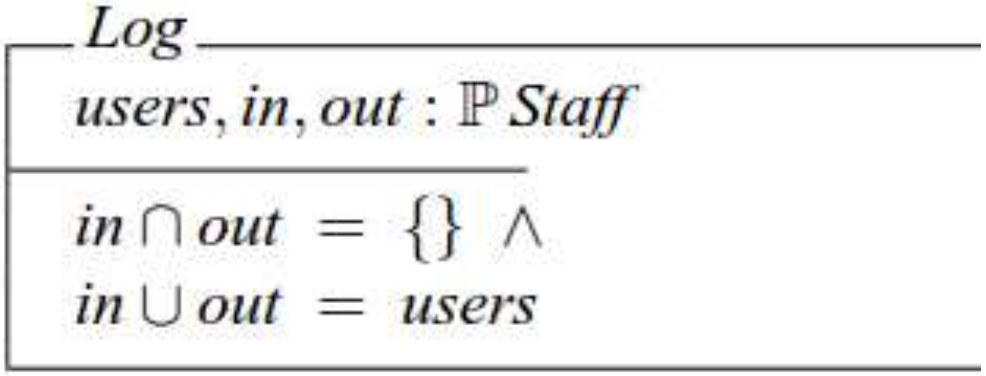
- We consider a specification of a system used to check staff members in and out of a building.

- Since we will be dealing with elements of type staff, we introduce the type Staff as a basic type:

[Staff]

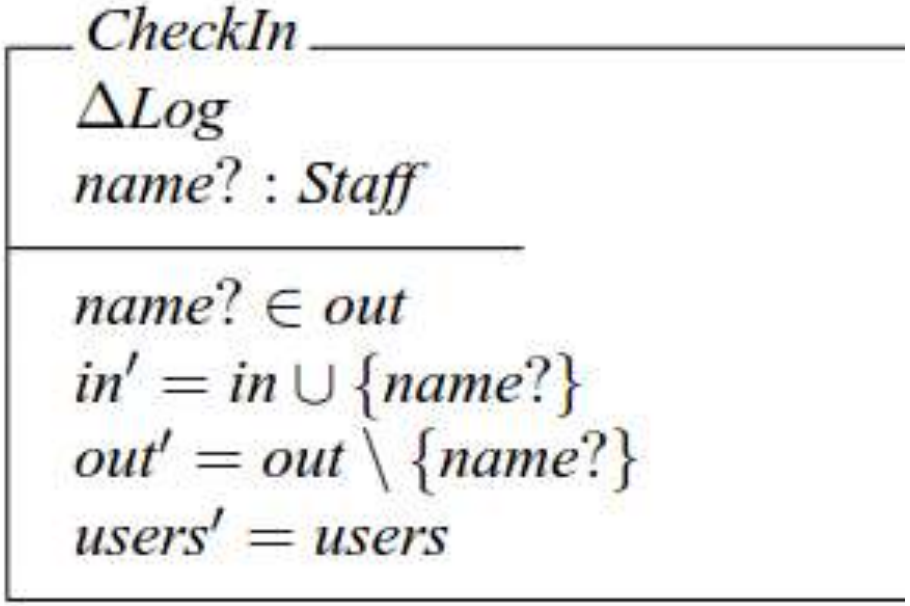
Check-In/Check-Out

- The state of the system is described by the following schema



Check-In/Check-Out

- An operation to check a staff member into the building is specified as follows



Check-In/Check-Out

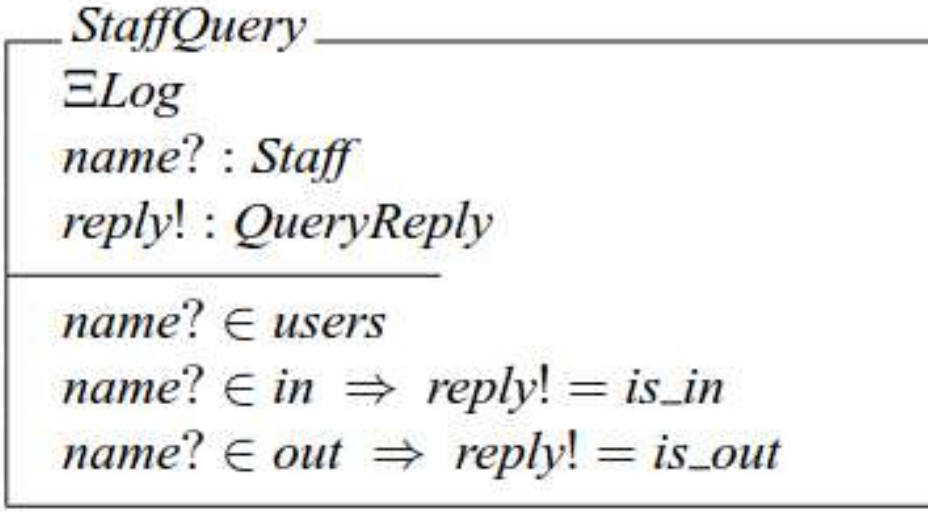
- Similarly, an operation to check a staff member out of the building may be specified as follows

$$
\begin{array}{|l}
\hline
\Delta Log \\
name? : Staff \\
\hline
name? \in in \\
out' = out \cup \{name?\} \\
in' = in \setminus \{name?\} \\
users' = users \\
\hline
\end{array}
$$

Check-In/Check-Out

- A query operation to check whether a particular member of staff is in or out will give an output parameter of the following type:

QueryReply == is in | is out



Check-In/Check-Out

- Typically the system would be initialised so that all sets are empty