Formal Method in Software Engineering (SE-313)

Course Teacher
Assistant Professor
Engr. Mustafa Latif

Introduction to Z Formal Specification

Z Formal Specification

Introduction

- **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.**
- **Theorem provers may be employed to demonstrate that the software implementation meets its specification.**

Z Formal Specification

Introduction

- **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(state space).**
- **The schemas may be used as building blocks and combined with other schemas.**

Z Formal Specification

Introduction

- 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.
- Exception handling is addressed by defining schemas for the exception cases.
- These are then combined with the original operation schema.
- Mathematical data types are used to model the data in a system, and these data types obey mathematical laws.
- These laws enable simplification of expressions and are useful with proofs.

5

Z Formal Specification

Introduction

- Operations are defined in a precondition/postcondition style.
- A precondition must be true before the operation is executed, and
- The postcondition must be true after the operation has executed. The precondition is implicitly defined within the operation.
- Each operation has an associated proof obligation to ensure that if the precondition is true, then the operation preserves the system invariant.
- The system invariant is a property of the system that must be true at all times.

6

Z Formal Specification

Introduction

```
–SqRoot––––––––
 num?, root! : ℝ
–––––
num? ≥ 0
root!² = num?
root! ≥ 0
–––––––––––––
```

Specification of positive square root

Z Formal Specification

Introduction

- The precondition for the specification of the square root function 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. (i.e. 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.
- The poststate of a variable being distinguished by priming the variable, e.g. v'.

Z Formal Specification

Introduction

- 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. (ℕ, ℤ and ℝ).
- The declaration of a variable x of type X is written x: X.
- It is also possible to create your own types in Z.

Z Formal Specification

Introduction

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

Z Formal Specification

Introduction

- For simple systems, direct refinement (i.e. one step from abstract specification to implementation) may be possible;
- In more complex systems, deferred refinement is employed, where a sequence of increasingly concrete specifications are produced to yield the executable specification.
- There is a calculus for combining schemas to make larger specifications.

11

Z Formal Specification

Introduction

- Example: The following is a Z specification to borrow a book from a library system. The library is made up of books that are on the shelf; books that are borrowed and books that are missing. The specification models a library with disjoint sets representing books on the shelf, on loan or missing. These are three mutually disjoint subsets of the set of books Bkd-Id (complete book set).

12

Z Formal Specification

Introduction

- The system state is defined in the Library schema, and operations such as Borrow and Return affect the state.
- The notation ℙ Bkd-Id is used to represent the power set of Bkd-Id (i.e. the set of all subsets of Bkd-Id).
- The disjointness condition for the library is expressed by the requirement that the pair-wise intersection of the subsets on-shelf, borrowed, missing is the empty set.

13

Z Formal Specification

Introduction

- The precondition for the Borrow operation is that the book must be available on the shelf to borrow.
- The postcondition is that the borrowed book is added to the set of borrowed books and is removed from the books on the shelf.

14

–*Library*————

on-shelf, missing, borrowed : $\mathbb{P}$ *Bkd-Id*

——

on-shelf $\cap$ *missing* = Ø
on-shelf $\cap$ *borrowed* = Ø
borrowed $\cap$ *missing* = Ø

————————

Specification of a library system

–*Borrow*————

Δ *Library*
b? :*Bkd-Id*

——

b? $\in$ *on-shelf*
on-shelf' = *on-shelf* \ {*b*?}
borrowed' = *borrowed* $\cup$ {*b*?}

————————

Specification of borrow operation

- In Z, we shall use three quantifers:

  $\forall$ — *the universal quantifier*;
  is read 'for all...'

  $\exists$ — *the existential quantifier*;
  is read 'there exists...'

  $\exists_1$ — *the unique quantifier*;
  is read 'there exists a unique...'

- The simplest form of quantified formula in Z is as follows:

  $$\mathit{quantifier\ signature} \bullet \mathit{predicate}$$

  where

  - *quantifier* is one of $\forall$, $\exists$, $\exists_1$;
  - *signature* is of the form

    $$\mathit{variable} : \mathit{type}$$

  - and *predicate* is a predicate.

- EXAMPLES.
  - $\forall x : Man \bullet Mortal(x)$
    'For all x of type *Man*, x is mortal.'
    (i.e. all men are mortal)
  - $\forall x : Man \bullet \exists_1 y : Woman \bullet MotherOf(x, y)$
    'For all x of type *Man*, there exists a unique y of type *Woman*, such that y is the mother of x.'
  - $\exists m : Monitor \bullet MonitorState(m, ready)$
    'There exists a monitor that is in a ready state.'
  - $\forall r : Reactor \bullet \exists_1 t : 100 \mathinner{..} 1000 \bullet Temp(r) = t$
    'Every reactor will have a temperature in the range 100 to 1000.'

19

- More examples:
  - $\exists n : \mathbb{N} \bullet n = (n * n)$
    'Some natural number is equal to its own square.'
  - $\exists c : EC \bullet Borders(c, Albania)$
    'Some EC country borders Albania.'
  - $\forall m, n : Person \bullet \neg Superior(m, n)$
    'No person is superior to another.'
  - $\forall m : Person \bullet \neg \exists n : Person \bullet Superior(m, n)$

20

Z Formal Specification

Schema

- The Z schema is a 2-dimensional graphical notation for describing:
  - state spaces;
  - operations.

SchemaName
Declarations
$Predicate_1; \cdots; Predicate_n$

or of the form

SchemaName
Declarations

21

Z Formal Specification

Schema

- Once introduced, SchemaName will be associated with the schema proper, which is the contents of the box.
- The declarations part of the schema will contain:
  - a list of variable declarations; and
  - references to other schemas (this is called schema inclusion).
  - Variable declarations have the usual form:
    - $x_1; x_2; : : : ; x_n : T;$
    - The predicate part of a schema contains a list of predicates, separated either by semi-colons or new lines.

22

Z Formal Specification

State Space Schemas

- Here is an example state-space schema, representing part of a system that records details about the phone numbers of staff.
- (Assume that NAME is a set of names, and PHONE is a set of phone numbers.)

$$\begin{array}{|l}
\textit{PhoneBook} \\
\hline
known : \mathbb{P}\, NAME \\
tel : NAME \rightarrowtail\!\!\!\!\!\!+\; PHONE \\
\hline
\operatorname{dom} tel = known \\
\hline
\end{array}$$

Z Formal Specification

State Space Schemas

- The declarations part of this schema introduces two variables: known and tel.
- The value of known will be a subset of NAME, i.e., a set of names. This variable will be used to represent all the names that we know about — those that we can give a phone number for.
- The value of tel will be a partial function from NAME to PHONE, i.e., it will associate names with phone numbers.

Z Formal Specification

State Space Schemas

- The declarations part is separated from the predicate part by the horizontal line.
- The predicate part contains the following invariant:
  - The domain of tel is always equal to the set known.

Z Formal Specification

Operation Schemas

- In specifying a system operation, we must consider:
  - the objects that are accessed by the operation, and of these:
    - ∗ the objects that are known to remain unchanged by the operation (cf. value parameters);
    - ∗ the objects that may be altered by the operation (cf. variable parameter);
  - the pre-conditions of the operation, i.e., the things that must be true for the operation to succeed;
  - the post-conditions — the things that will be true after the operation, if the pre-condition was satisfied before the operation.

Z Formal Specification

Operation Schemas

- **Return to the telephone book example, and consider the 'lookup' operation: we put a name in, and get a phone number out.**
  - **this operation accesses the PhoneBook schema;**
  - **it does not change it;**
  - **it takes a single 'input' — a name for which we want to find a phone number;**
  - **it produces a single output — a phone number.**
  - **it has the pre-condition that the name is known to the database**

27

$$
\begin{array}{|l}
\hline
\textit{Find} \\
\Xi \textit{PhoneBook} \\
\textit{name}? : \textit{NAME} \\
\textit{phone}! : \textit{PHONE} \\
\hline
\textit{name}? \in \textit{known} \\
\textit{phone}! = \textit{tel}(\textit{name}?) \\
\hline
\end{array}
$$

28

Z Formal Specification

Operation Schemas

- This illustrates the following Z conventions:
  - placing the name of the schema in the declarations part ‘includes’ that schema — it is as if the variables were declared where the name is;
  - ‘input’ variable names are terminated by a question mark; the only input is name?
  - ‘output’ variables are terminated by an exclamation mark; the only output is phone!

29

Z Formal Specification

Operation Schemas

- This illustrates the following Z conventions:
  - the Ξ (Xi) symbol means that the PhoneBook schema is not changed;
  - if we have written a Δ (delta) instead of Ξ, it would mean that the PhoneBook schema did change.
  - the pre-condition is that name? is a member of known;
  - the post-condition is that phone! is set to tel(name?).

30

Z Formal Specification

Operation Schemas

- Here is another schema: this one add's a name/phone pair to the phone book.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AddName}} \\
\hline
\Delta \textit{PhoneBook} \\
\textit{name}? : \textit{NAME} \\
\textit{phone}? : \textit{PHONE} \\
\hline
\textit{name}? \notin \textit{known} \\
\textit{tel}' = \textit{tel} \cup \{\textit{name}? \mapsto \textit{phone}?\} \\
\hline
\end{array}
$$

31

Z Formal Specification

Operation Schemas

- This illustrates the following Z conventions:
  - This schema accesses PhoneBook and does change it (hence the use of Δ rather that Ξ.)
  - Two inputs: a name (name?) and phone number (phone?).
  - Pre-condition: the name is not already in the database.

32

Z Formal Specification

Operation Schemas

- This illustrates the following Z conventions:
  - Post-condition: tel after the operation is the same as tel before the operation with the addition of maplet name? $\mapsto$ phone? (name?,phone?)(The maplet arrow provides alternate syntax without parentheses).
  - Appending a ′ to a variable means 'the variable after the operation is performed'.