MUST

Wisdom & Virtue

MIRPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY (MUST), MIRPUR
DEPARTMENT OF SOFTWARE ENGINEERING

Formal Methods in Software Engineering

Lecture [13]: Formal Specification through Z Language

Engr. Samiullah Khan

(Lecturer)

Topics discussed in Today's Lectures

- Formal Specification
- Z Language

Formal Specification

- Formal specifications use mathematical notation to describe in a precise way the properties which an information system must have
- They describe what the system must do without saying how it is to be done
- This abstraction makes formal specifications useful in the process of developing a computer system
  - Because they allow questions about what the system does to be answered confidently

Formal Specification for Investigation

- A formal specification can serve as a single, reliable reference point for those:
  - who investigate the customer's needs
  - who implement programs to satisfy those needs
  - who test the results
  - who write instruction manuals for the system
- Because it is independent of the program code, a formal specification of a system can be completed early in its development.

Formal Specification through Z Language

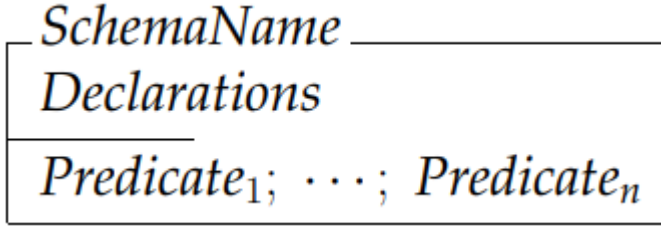
- Z is a way of decomposing a specification into small pieces called **schemas**
- By splitting the specification into schemas, we can present it piece by piece
- Each piece can be linked with a commentary which explains informally the significance of the formal mathematics
- In Z, schemas are used to describe both static and dynamic aspects of a system

Formal Specification through Z Language

- The **static aspects** include:
  - the states it can occupy
  - the constant relationships that are maintained as the system moves from state to state.
- The dynamic aspects include:
  - the operations that are possible
  - the relationship between their inputs and outputs
  - the changes of state that happen

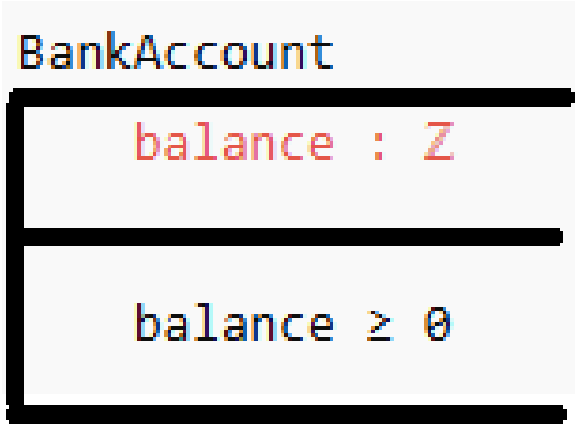
Z Schema in Z Language

- A schema is written as:



- **Simple Z Schema Examples**

1- Schema for a Simple Bank Account
Purpose: Represents the state of a bank account.



Explanation:
- balance is an integer (ℤ)
- Must always be **non-negative**

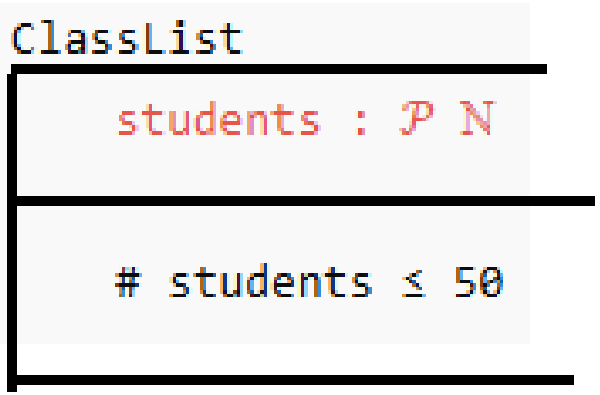
Z Schema in Z Language – Examples

2. Schema for Adding a Student to a Class List

Purpose: Maintain a set of student roll numbers.

Explanation:

- **students** is a set of natural numbers
- Maximum 50 students allowed



3. Schema for a Login System

Purpose: Store username and password constraints.

Explanation:

- username and password are sequences of characters
- Username must be at least **3 characters long**
- Password must be at least **6 characters long**

Z Schema in Z Language

- Z Schema (Z Notation) is a formal specification structure used in Z language
- It is based on set theory and first-order predicate logic
- A schema describes a state or an operation of a system using:
  - Declarations (variables and their types)
  - Predicates (constraints or relationships between variables)

Z Language - Example

- Some numbers don't have integer square roots
- What happens if you call **iroot** with a set to 3?
- Negative numbers don't have square roots at all
- What if you call **iroot** with a set to -4?
- Does it return anything, or does it loop forever — or crash?
- These questions reveal the problem with the name
- They don't explain how the function behaves for every input
- Can't we come up with a better description?

```
int iroot(int a)
/* Integer square root */
{
   int i,term,sum;

   term=1; sum=1;
   for (i=0; sum <= a; i++)
       term=term+2;
       sum=sum+term;
   }
   return i;
}
```

Z Language - Example

- The code is supposed to compute the integer square root
- So you might guess that if you call **iroot** with a set to 4, it should return 2
- If you type in the code, write a driver program, and run a little test, you will find that it does

```
int iroot(int a)
/* Integer square root */
{
   int i,term,sum;

   term=1; sum=1;
   for (i=0; sum <= a; i++)
        term=term+2;
        sum=sum+term;
   }
   return i;
}
```

Z Language - Example

- We need more than just the code
- We need a specification
- The code describes the computation itself, but a specification describes the result of the computation
- We start with a definition of square root in English
- When you multiply the square root of a number by itself, you get the original number back again
- We can say the same thing more concisely in mathematical notation:

$$\sqrt{a} \times \sqrt{a} = a$$

```
int iroot(int a)
/* Integer square root */
{
   int i,term,sum;

   term=1; sum=1;
   for (i=0; sum <= a; i++)
        term=term+2;
        sum=sum+term;
   }
   return i;
}
```

Z Language - Example

We can do it easily in Z. Here is a specification for `iroot`, expressed in a single *Z paragraph* called an *axiomatic definition*:

$$iroot : \mathbb{N} \rightarrow \mathbb{N}$$

$$\forall a : \mathbb{N} \bullet$$
$$iroot(a) * iroot(a) \leq a < (iroot(a) + 1) * (iroot(a) + 1)$$

The first thing you will notice is that the Z paragraph is set off from the surrounding prose by lines that suggest a sort of box. Z turns the commenting convention of programming languages inside out: In Z we delimit the formal text, not the prose. There is no way to write informal comments inside a Z paragraph. You are supposed to provide the explanation in the surrounding text, and keep the Z parts brief enough that your readers don't get lost.

```
int iroot(int a)
/* Integer square root */
{
  int i,term,sum;

  term=1; sum=1;
  for (i=0; sum <= a; i++)
      term=term+2;
      sum=sum+term;
  }
  return i;
}
```

THANKS