1. State-based Specification Method

Features

- The description of the system behavior is centered around the notion of state transition.
- The operations (functions) of the system are specified by description how their execution change the state of the system.

State-based Specification Method

- **Steps**
  - Define the variables that give states of the target system and any invariant properties relating those variables.
  - Define an initial operation to set the values of the variables to some suitable initial state that satisfies the invariant requirement.
  - Define operations on a state that change that state while maintaining the invariant properties.
  - Define enquiries that obtain information about the system without changing its state.

Types in Z

- **Typed set theory**
  - All possible members of a typed set are considered to have something in common, and are said to have the same type.
  - The notion of type helps in two ways: it avoids certain mathematical paradoxes; it allows checks to be made that statements about sets make sense.
  - Z is based on typed set theory.

Types in Z

- **The built-in type Integer**
  - The built-in type Integer can be used in Z without the need to introduce it explicitly.
  - *Z* (ZZ): Pronounced 'integer' of 'zed-zed' or 'fat Z'.
  - Operation on *Z* : +, -, *, div, mod
  - The range of values m .. n stands for the set of integers m to n inclusive. m .. n = $\varnothing$ if m > n.

Types in Z

- **The standard type Natural**
  - **The set Natural is a standard subset of the type Integer that adds the constraint that the value must always be nonnegative.**
  - $\mathbb{N}$ **(NN):Pronounced 'natural' or 'en-en' or 'fat N'.**
  - $\mathbb{N}_1$**: the set of nature numbers excluding zero.**
- **Numerical relations**
  - $=. \neq, <, \leq, >, \geq$

Types in Z

- **Basic (Given) types**
  - The basics (given) types of a specification are declared without concern for how their actual elements are to be represented.
  - A basic type is declared by writing its name in square brackets, with a comment to indicate its intended meaning.
  - Ex: [PERSON]   the set of all persons
- **Free types**
  - A type introduced by listing the identifiers of its elements.
  - freeType ::= $element_1$ | $element_2$ | … $element_n$

Variables in Z

- **Declarations of variables**
  - Each variable name designing a value must be declared.
  - That means it must be introduced and the type of value it refers to must be stated.
  - v:TYPE, pronounced ‘v is one of the set of values TYPE’ or ‘v is drawn from the set TYPE’ or ‘v is a TPYE’.
- **Ex.**
  - chauffeur: PERSON

Sets in Z

- **Values of sets**
  - The value of a set can be written by listing its values within braces ('curly brackets').
- **Validity of membership test**
  - The value to be tested for membership must be an element of the underlying type of the set, otherwise, the test is neither true nor false but illegal.
- **Size (cardinality) of a set**
  - The number of elements in a set is called its size (cardinality), and is signified by hash sign.
  - $\# \varnothing = 0$

Sets in Z

- **Powersets**
  - The powerset of a set S is written as $\mathbb{P}$S.
  - # $\mathbb{P}$S = $2^{\#S}$
- **Difference**
  - The difference of two sets is the set containing all those elements of the first set that are not in the second set.
  - S \ T

Sets in Z

- **Distributed union and intersection**
  - The distributed union of a set of sets is the set containing just those elements that occur in at least one of the component sets.
  - The distributed intersection of a set of sets is the set containing just those elements that occur in all of the component sets.
- **Partition**
  - A set of sets is said to partition a set S if the sets are disjoint and their distributed union is the set S.

Example: Using Sets to Describe a System

- **Scenario**
  - The passengers aboard an aircraft.
- **Constraints**
  - No seat numbers, first-come-first-served, fixed capacity.
- **Assumptions**
  - People are identified uniquely.
- **Basic type**
  - [PERSON] the set of all possible uniquely identified persons
- **Variables**
  - capacity: $\mathbb{N}$ the seating capacity of the aircraft
  - onboard: $\mathbb{P}$PERSON the state of the aircraft system

Example: Using Sets to Describe a System

- **Invariant property**
  - $\# \text{onboard} \leq \text{capacity}$
- **Initialization operation**
  - The value of a variable after an operation is denoted by its name ‘decorated’ with a prime sign.
  - $\text{onboard}' = \emptyset$
- **Boarding operation**

  p: PERSON

  $p \notin \text{onboard}$

  $\# \text{onboard} < \text{capacity}$

  $\text{onboard}' = \text{onboard} \cup \{p\}$

Example: Using Sets to Describe a System

- **Disembark operation**

  p: PERSON

  $p \in$ onboard

  onboard' = onboard \ {p}

- **Enquiries**
  - **Number on board**

    numOnboard: $\mathbb{N}$

    numOnboard = # onboard

    onboard' = onboard
  - **Person on board**

    RESPONSE ::= yes | no

    p: PERSON

    reply: RESPONSE

    (($p \in$ onboard and reply = yes) or ($p \notin$ onboard and reply = no))

    onboard' = onboard

Example: Using Sets and Logic Operators to Describe a System

$\text{RESPONSE} ::= \text{OK} \mid \text{twoErrors} \mid \text{onBoard} \mid \text{full} \mid \text{notOnBoard}$

⌗ Onboard operation

p: PERSON

reply: RESPONSE

$(p \notin \text{onboard} \wedge \#\text{onboard} < \text{capacity} \wedge$

$\text{onboard}' = \text{onboard} \cup \{p\} \wedge \text{reply} = \text{OK})$

$\vee$

$(p \in \text{onboard} \wedge \#\text{onboard} = \text{capacity} \wedge$

$\text{onboard}' = \text{onboard} \wedge \text{reply} = \text{twoErrors})$

$\vee$

$(p \in \text{onboard} \wedge \#\text{onboard} < \text{capacity} \wedge$

$\text{onboard}' = \text{onboard} \wedge \text{reply} = \text{onBoard})$

$\vee$

$(p \notin \text{onboard} \wedge \#\text{onboard} = \text{capacity} \wedge$

$\text{onboard}' = \text{onboard} \wedge \text{reply} = \text{full})$

Example: Using Sets and Logic Operators to Describe a System

Disembark operation

p: PERSON

reply: RESPONSE

$(p \in onboard \wedge onboard' = onboard \setminus \{p\} \wedge$

$reply = OK)$

$\vee$

$(p \notin onboard \wedge onboard' = onboard \wedge$

$reply = notOnBoard)$