N notation

一、概述

1.Z 是一种基于集合理论和一阶谓词逻辑的形式化语言，英国牛津大学的程序设计研究小组（PRG）在 20 世纪 80 年代初。 ISO 的国际标准化 Z 工作于 2002 年完成

2.两种类型的抽象

表示抽象：数据从数据结构的表示细节中抽象，使用关系、函数、集合、序列、包等的数据类型，并构成 Z 的类型系统。

操作抽象：描述在表示抽象中所引入的抽象算法或操作。

在 Z 规范中，表示抽象通过类型定义、全局变量以及以及状态空间声明来表示，操作抽象通过函数和基于一阶谓词逻辑的操作表示。

二、Schema(模式)

1.Z 规范的基本描述单位。

一个软件系统的 Z 规范主要由若干个模式构成。

模式描述系统的静态和动态的行为，从软件系统的状态和状态之间的转化两个方面来实施软件系统的描述。

2.模式类型

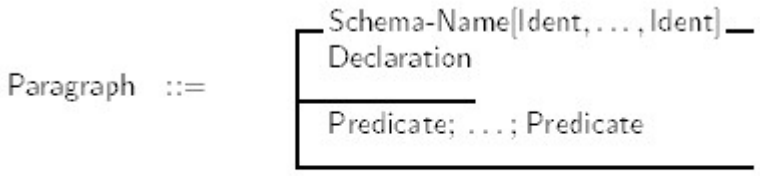
模式分为两类：

状态模式：定义目标软件系统某一部分的状态及其约束特性，它通过相应的抽象数据类型的变量以及在它们上面所定义的一些约束关系来进行描述。

操作模式：描述系统某部分的行为特征，它通过描述在操作之前的该部分的状态值与操作之后的该部分的状态值之间的关系，来定义该部分的某种操作的特性

3.模式的结构

一个模式有一个唯一的标识即模式名称，并包含一个声明（Declarations）部分和一个断言(Predicates)部分。



Paragraph ::= Schema-Name[Ident, . . . , Ident] ≙ Schema-Exp

声明（Declarations）部分：引入某些类型的变量，这些变量为模式的局部变量.

断言(Predicates)部分：描述在这结局部变量之间，或者局部变量与在该模式之前声明的全局变量（常量）之间的不变式(invariant)关系。并定义模式一些运算，通过这些运算，从简单的模式构造出复杂的模式。

4.非形式描述部分

为了保证目标系统描述的可读性和可理解性，还需要安排和使用非形式的描述，在整个规范文档中，非形式的描述在文档的开始部分，给出问题的描述、系统的需求、对系统所作的假设，以及在系统中使用的术语、词汇等。

5.Z 规范的文档结构：

（1）给定类型的声明和全局变量（常量）的定义：它构成声明（Declarations）部分中所使用的类型的基础。

（2）系统的抽象状态的描述：使用状态模式来对系统的状态空间及其约束特性进行描述。

（3）系统的初始化定义：对系统的初始状态进行描述，即对系统的状态模式中的变量赋予初值。

（4）对正常条件下系统操作的定义：正常条件－指系该操作能成功执行的条件

（5）计算已经定义的操作的前置条件：对于一个操作，总需要满足一定的前置条件才能正常执行相应的操作，对于每个操作模式，写出其前置条件模式，并对该前置条件模式进行简化。

（6）描述完整的操作和（在可能的出错情况下对操作的定义）：一个正常条件下的操作的前置条件指出该操作知什么条件下可以执行，而对于这些条件的否定称为出错条件。为了保证系统在所有的情况下都有相应的动作，就需要对“非正常”条件下的操作也给出描述。

（7）对规范中的一些性质进行证明：能对有关性质进行形式证明是 Z 规范的一个很重要的优点，它能保证正确性和一致性，此外还需要对初始化进行证明。

(8)建立规范的摘要和目录：为使文档容易阅读和修改。

三、表示抽象

1.集合、关系和函数

集合的基本类型操作（∈、∪、∩、 / 、# 等等）

幂集

元组和笛卡儿积：二元组（序偶）(x，y）也记为：$x \mapsto y$

关系：

$\leftrightarrow$	$X \leftrightarrow Y$	X与Y之间的二元关系，即 $P(X \times Y)$
$\mapsto$	$x \mapsto y$	
dom	dom R	关系R的定义域
ran	ran r	关系R的值域
$\fatsemi$	$R_1 \fatsemi R_2$	关系复合
$\bigcirc$	$R_1 \bigcirc R_2$	反向关系复合
$\lhd$	$S \lhd R$	定义域限制
$\rhd$	$R \rhd S$	值域限制
$\lhd\!\!\!-$	$S \lhd\!\!\!- R$	定义域反限制
$\rhd\!\!\!-$	$R \rhd\!\!\!- S$	值域反限制

函数

$\pfun$	部分函数（偏函数）	X 到 Y 的一个部分函数是 X 与 Y 之间的一个关系，它将其定义域 X 中部分元素映射到其值域 Y 中的一个元素。（从类型 X 到类型 Y 的所有偏函数的集合，一个偏函数是不必在定义域类型的所有元素定义的函数。）
$X \pfun Y$	X 到 Y 的所有部分函数的集合	
$\fun$	全函数	X 到 Y 的全函数是一个特殊的部分函数，其定义域为整个源集合 X。（从类型 X 到类型 Y 的所全函数的集合，一个全函数是在定义域类型的所有元素定义的函数。） $X \fun Y == \{ f : X \fun Y \mid \mathrm{dom}\ f = X \}$
$X \fun Y$	X 到 Y 的所有全函数的集合	
$\finfun$	有限函数	X 到 Y 的一个有限函数是 X 到 Y 的部分函数，其定义域是 X 的一个有限子集。
$X \finfun Y$	X 到 Y 的所有有限函数的集合	
	入射函数（injective）	从类型 X 到类型 Y 的所有偏函数、并且其反函数是一个从 Y 到 X 的偏函数的集合。
$X \rightarrowtail Y$	全入射函数	
$X \rightarrowtail Y$	部分入射函数	
$X \finfun Y$	有限入射函数	
	满射函数	满射函数是值域是整个 Y 的一个 X 到 Y 的函数。
$X \twoheadrightarrow Y$	全满射函数	
$X \twoheadrightarrow Y$	部分满射函数	
	双射函数	双射函数是一个 X 到 Y 的一一对应函数
$X \rightarrowtail Y$	全双射函数	
$X \rightarrowtail Y$	部分双射函数	

2.序列、包

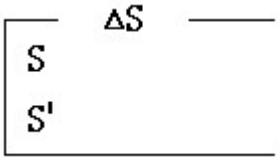
序列是带有次序的对象的集合描述，用< >表示，**seq X** 是一个类型，声明一个序列类型的

对象，该对象中的元素是集合 **X** 中的元素。序列可看成是从 N 到 X 的一个有限函数。
包（**bag**）不仅描述一个数据对象的集合，而且还包含了其中每个对象所出现的次数，包与集合相同之处是元素的出现次序是无关紧要的;不同之处是每个元素可以重复出现多次，并且每个对象的出现次数是有意义的。**bag** 用〖〗表示。**Bag X** 表示包类型，由集合 X 中的元素所构成的包可看成是从 X 到 N 的一个部分函数
L=〖Tom, John, John, Tom, Bob〗,包 L 等价于：

$$\{\text{Tom} \mapsto 2,\ \text{John} \mapsto 2,\ \text{Bob} \mapsto 1\}$$

3.自由类型
类型的声明和全局变量（常量）的定义

四、操作抽象
操作抽象在表示抽象的基础上描述了对抽象数据所进行的操作及抽象算法,主要可分为两类：操作模式和函数。操作模式描述了对系统所进行的操作，它直接对状态空间的变量活产生影响;而函数描述了独立于系统状态的某个行为，并不访问具体的系统变量。
1.模式的扩充
S;D:对模式 S 的说明部分扩充，即加上新的说明部分 D.
S|P:对模式 S 的谓词部分扩充，即加上新的谓词 P.
2.模式包含
3.模式变量换名
S[y1/x1,y2/x2,…,yn/xn]
将模式 S 中的说明变量 xi 用 yi 替换。
4.模式变量的消除
S\(x1,x2,…,xn
5.模式的修饰
模式的修饰就是在一个模式名之后加上“ ’ ”来对该模式进行修饰。模式修饰的作用是将修饰“ ’ ”应用到在该模式中所声明的所有变量上，即，对声明部分的所有变量，以及这些变量在谓词部分的所有出现都进行修饰，修饰的模式称为后状态模式。
Δ和Ξ表示两种常用的模式的简写：
ΔS 定义为



ΞS 定义为对模式 S 进行一种操作后得到的后状态模式 S’与 S 是相同的，即该操作引起系统状态的任何变化。
6.命题的联结词
两个模式，可以用逻辑连接词：∧、∨、¬ 等组成新模式。
7. 前置条件模式（Pre）和初始状态模式(Init)
前置模式 Pre 用以将一个操作模式的前置条件从该模式中分离出来,而隐藏了该模式的后状态变量和输出变量，运算的结果仍一个模式

五、Z 规范的一个应用

自动售货机系统的规范描述

需求说明：

一台自动售货机可接受的硬币：1 元，5 角、1 角、5 分、1 分;

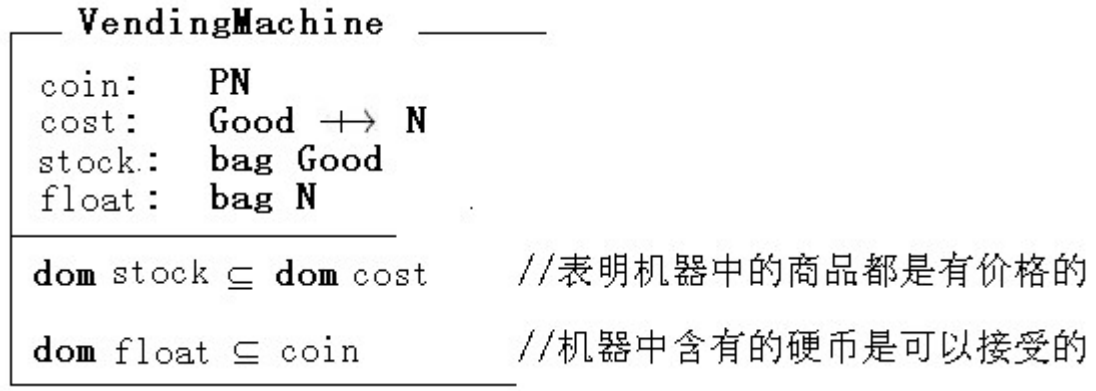
货物价格以分为单位计算;

1.自由类型

[Good]

[Report]

2.售货机的状态说明模式



这里 coin 是所有可接受有硬币的集合，以分值标记，函数 cost 返回的是以分值表示的该商品的价格：如 cost(pencil)=20, cost(note)=50

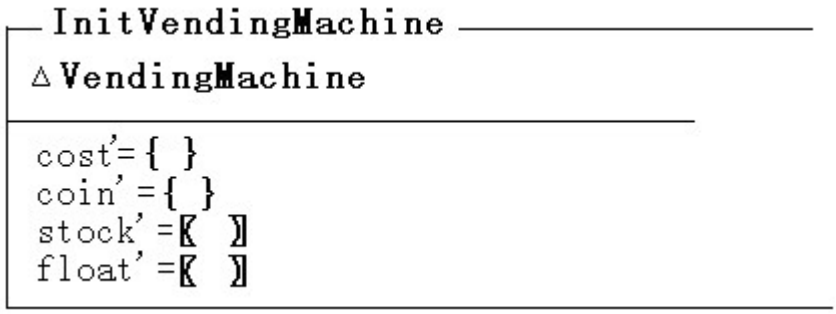
包 stock 记录了当前机器中每一种商器的数目。如

$stock=\{ pencil \mapsto 10,\ note \mapsto 15\ , ...\}$

包 float 记录了当前机器中各种硬币的数目。如

$float=\{ 100 \mapsto 2,\ 50 \mapsto 10, 10 \mapsto 30,\ ...\}$

3 .初始化模式



4 .操作模式 1 －－为一种商品定价(**Price**)

如果商品已经在机器中，该操作还可作为修改商品价格。

Price

ΔVendingMachine
item?: Good
price?: $\mathbb{N}$

$cost' = cost \oplus \{ item? \mapsto price? \}$
$coin' = coin$
$stock' = stock$
$float' = float$

模式 **Success** 描述操作成功时应给出的报告：

Success

rep!: Report

---

rep! = "OK"

这样商品定价操作的完整表示：

$$doPrice := Price \wedge Success$$

5.可接受硬币操作 **Accept** 模式

Accept

ΔVendingMachine
c?: $\mathbb{N}$

$c? \notin coin$
$coin' = coin \cup \{ c? \}$
$cost' = cost$
$stock' = stock$
$float' = float$

Accept 模式的前置条件：c？在 coin 时处理。

Accept 模式的前置条件模式 **AlreadyAcceptable** 模式

AlreadyAcceptable

ΞVendingMachine
c?: $\mathbb{N}$
rep!: Report

$c? \in coin$
rep! = "coin already acceptable"

接受一种硬币的操作模式：

$$doAccept := (Accept \wedge Success) \vee AlreadyAcceptable$$

6.添加商品到机器中的 **Restock** 操作模式：

Restock

Δ VendingMachine

new? : bag Good

---

dom new? $\subseteq$ dom cost

stock' =stock $\uplus$ new?　// $\uplus$ 包运算符之一，B1 $\uplus$ B2 表示生成一个新的包包中每个对象出现的次数是在B1中出现次数与B2中出现次数的和

coin' =coin

float' =float

cost' =cost

Restock 操作模式的前置条件模式 **GoodsNotPrice**：

GoodsNotPrice

Ξ VendingMachine

new? : bag Good

rep! : Report

---

dom new? $\not\subseteq$ dom cost

rep! ="Some goods are unpriced!"

Restock 操作的完整规范为：

$$doRestock := (Restock \wedge Success) \vee GoodsNotPriced$$

7 .顾客购买模式 **Buy**

在 **Buy** 模式中，有声明变量 in?表示顾客投入机器的硬币，变量 out！表示机器的找零输出的硬币,item!表示顾客实际买的商品，为此需要有一个函数 **sum,**计算一个数的包的和，例如：

$$sum(\{1 \mapsto 8, 50 \mapsto 3\} = 1*8+50*3=158$$

sum[L]

sum: bag N $\longrightarrow$ N

$\forall$ i, j:N; L:bag N

sum [[]] =0 $\wedge$ sum({ i $\mapsto$ j} $\cup$ L)=i*j+sum L

Buy

ΔVendingMachine
in?, out! : bag N
item! : Good

---

item! $\in$ dom stock　　//前置条件，如果商品在机器中，可以购买
dom in? $\subseteq$ coin　//前置条件，只有可接受的硬币，才可以购买
sum(in?)>= cost(item!) //前置条件，顾客投入足够的硬币金额
out! $\subseteq$ float　//前置条件，如果机器中确实有，足够找零
sum(in?)=sum(out!)+cost(item!) //机器找零
float' $\uplus$ out!=float $\uplus$ in? //机器中的钱的增加量等于顾客所买商品的所花的钱数
stock' $\uplus$ { item! $\mapsto$ 1} = stock　//顾客向机器买的商品从机器中送出
coin' =coin
cost' =cost

Buy 操作模式在四个前置条件模式：

NotInstock

ΞVendingMachine
item! : Good
rep! : Report

---

item! $\notin$ dom stock
rep! ="Item not in stock!"

ToolittleCoin

ΞVendingMachine
in? : bag N
item! : Good
rep! : Report

---

sum(in?) < cost(item!)
rep! = "Insert more coins"

ChangesUnavailable

Ξ VendingMachine

in?, out! : bag N

item! : Good

rep! : Report

$\neg \exists L : \text{bag } N \mid (L \subseteq \text{float} \wedge \text{sum}(in?) = \text{sum}(L) + \text{cost}(item!))$

rep! = "Change Unavailable!"

UnacceptCoin

Ξ VendingMachine

in? : bag N

rep! : Report

$out! \notin \text{float}$

rep! = "Unacceptable coin"

购买商品操作的完整规范：

$$doBuy := (Buy \wedge Success) \vee NotInstock \vee UnacceptCoin \vee ChangesUnavailable \vee ToolittleCoin$$