Standard for The Set Programming Language

Jedi Yang

I. Data Structure

The Set Programming Language aims at using sets and assist with vectors to build a Functional Programming Language. The data structure of Set is more important than language and grammar.

A. Atom

Atom is the fundamental unit of Set. It means a set with only one number, either an integer or a decimal, but only numbers are stored in atoms.

An atom can be expressed as $\{A\}$, but it can also be simplified as A. However, these two expression are different when doing calculation:

$\{A\} + \{B\} == \{A, B\}$ while $A + B$ is the sum of number A and B.

An Atom with brackets around accept Set Operations while an atom without brackets accept Basic Operations (see Operations)

1) Number: Due to IEEE 754, if an integer is stored in a float or double, it is almost impossible to determine whether the number is an integer. The numbers in Set Programming Language use Irreducible Fractions to store both integers and decimals. An irreducible fraction contains two integers, a denominator and a numerator. Atoms can only store Irreducible Fractions.

2) Boolean: Boolean is special number with true and false only. Typically an atom of 0/0 is false. A set containing value 0/0 is also false. Otherwise, valid numbers and sets are seen as true including zero. During comparison, true is returned as 1/1 and false is 0/0.

B. Set

Set is a new data structure that is rare in other programming languages. It is similar to an Array, but there are two restrictions:

1) A set has no two identical numbers
2) A set is ordered. The smallest is the head of the set.

If a set does not meet the restrictions, identical numbers should be reduced and values are sorted automatically. Set can be catagorized into Enumerative Sets and Conditional Sets.

1) Enumerative Sets: This branch contains sets such that all numbers within the sets are listed. Enumerative Sets are stored in linked lists, with the smallest number is the head. All enumerative sets are finite.

For example: $\{1, 3, 5, 17/2, 10\}$ is an enumerative set.

2) Conditional Sets: This branch contains sets such that numbers are determined by an operation and conditions. Not all conditional sets are infinite, some of them can be listed out.

For example: $\{x : 2x + 7 | 2 < x < 7\}$ is a conditional set.

C. Vector

Vector is another name for Array. However, Vectors should accept vector operations such as dot product and it only accepts Numbers. Vectors and Sets are similar. Vectors are also stored in linked lists. The difference between Vectors and Sets are that vectors do not have restrictions for sets:

1) A vector can have two identical numbers
2) A vector can be randomly sorted.

There is a special form of Vector. Transformation Vector.

1) Transformation Vector: This is vector that describes a series of transformation to one or more than one variables. In transformation vectors, Functions are stored. The return values of the prior function acts as the arguments for the next one.

2) Function: It is similar to Lambda Expressions in other programming languages. A function can only support one line of code. Except the return values and arguments of a function, the function cannot use other variables. Functions are better used in transformation vectors to support complex operations.

D. String

String is a special vector, similar to most programming languages. However, it should have a different implementation using consequtive memory spaces.

II. Operation

A. Numbers

Unary Operation

1) -x

Binary Operation:

1) x+y
2) x-y
3) x*y
4) x/y
5) x%y
6) x&y
7) x|y
8) x==y
9) x!=y
10) x¿y
11) x¡y
12) x¿=y
13) x¡=y
14) x&&y
15) x||y

B. Set & Number

Binary Operation:

1) s+x == s.append(x)
2) s-x == s.remove(x)
3) s*x == s.forEach(this∼this*x)
4) s/x == s.forEach(this∼this/x)
5) s%x ==s.forEach(this∼this%x)

C. Sets

Binary Operation:

1) u+v == v.forEach(u.append)
2) u-v == v.forEach(u.remove)
3) u*v == v.forEach(u.exist)
4) u/v == (u+v).forEach(xnor(u.exist,v.exist))