

Sets and functions

Philippos Apolinario Costa

1 Sets

A set is a collection of things. You certainly know that collections do not have repeated items. I mean, if a guy or gal has a collection of stamps, s/he does not want to have two copies of the same stamp in his/her collection. If s/he has a repeated item, s/he will trade it for another item that s/he lacks in his/her collection. It is possible that if your father has a collection of ancient Greek coins, he will be willing to accept another drachma in his set. However, the two drachmas are not exactly equal; the left eye of the owl may have a scratch. Mathematicians collect other things, besides coins, stamps, and slide rules. They collect numbers, for instance; therefore you are supposed to learn a lot about sets of numbers.

\mathbb{N} is the set of natural integers. Here is how mathematicians write the elements of \mathbb{N} : $\{0, 1, 2, 3, 4 \dots\}$. In Clean, you may specify \mathbb{N} using the Zermelo-Frankel notation $[0, 1..]$, which means that \mathbb{N} starts with 0 and goes to infinity.

\mathbb{Z} is the set of integers, i.e., $\mathbb{Z} = \{\dots - 3, -2, -1, 0, 1, 2, 3, 4 \dots\}$. Why the set of integers is represented by the letter \mathbb{Z} ? I do not know, but I can make an educated guess. The set theory was discovered by Georg Ferdinand Ludwig Philipp Cantor, a Russian whose parents were Danish, but who wrote his *Mengenlehre* in German! In German, integers may have some strange name like *Zahlen*.

You may think that set theory is boring; however, many people think that it is quite interesting. For instance, there is an Argentinean that scholars consider to be the greatest writer that lived after the fall of Greek civilization. In few words, only the Greeks could put forward a better author. You probably heard Chileans saying that Argentineans are somewhat conceited. *You know*

what is the best possible deal? It is to pay a fair price for Argentines, and resell them at what they think is their worth. However, notwithstanding the opinion of the Chileans, Jorge Luiz Borges is the greatest writer who wrote in a language different from Greek. Do you know what was his favorite subject, tales? It was the Set Theory, or *Der Mengenlehre*, as he liked to call it.

When a mathematician wants to say that an element is a member of a set, he writes something like

$$3 \in \mathbb{Z}$$

If he wants to say that something is not an element of a set, for instance, if he wants to state that -3 is not an element of \mathbb{N} , he writes:

$$-3 \notin \mathbb{N}$$

Let us summarize the notation that Algebra teachers use, when they explain set theory to their students.

Double backslash. If a mathematician wants to buy the set of x^2 , such that x is a member of \mathbb{N} , he places the following order: $\{x^2 \setminus x \in \mathbb{N}\}$. If it is his lucky day, he will receive a package with the set

$$\{0, 1, 4, 9, 16, 25 \dots\}$$

Guard. If you want to say that x is a member of \mathbb{N} on the condition that $x > 10$, you can write $\{x \setminus x \in \mathbb{N} | x > 10\}$. The vertical bar is called guard, and introduces a condition.

Conjunction. In Mathematics, you can use a symbol \wedge to say **and**; therefore $x > 2 \wedge x < 5$ means that $x > 2$ and $x < 5$. There are people who prefer a double *et*; they will write the prior expression as $x > 2 \ \&\& \ x < 5$.

Disjunction. There are also two notations for **or**. Both expressions

$$(x < 2) \vee (x > 5)$$

and $(x < 2) || (x > 5)$ mean $x < 2$ or $x > 5$.

Using the above notation, you can define the set of rational numbers:

$$\mathbb{Q} = \left\{ \frac{p}{q} \setminus p \in \mathbb{Z}, q \in \mathbb{Z} | q \neq 0 \right\}$$

In informal English, this expression means that a rational number is a fraction

$$\frac{p}{q}$$

such that p is a member of \mathbb{Z} and q is also a member of \mathbb{Z} , submitted to the condition that q is not equal to 0.

Clean does not have a notation for sets, but you can use lists to represent sets. For instance, choose the option **File/New File** from the task menu. In the New File dialog, create the file `playset.icl` that will hold the main program.

```
module playset
import StdEnv
```

```
ZI = [-2..2]
```

```
Start= [(p, q) \ \ p <- ZI, q <- ZI | q <> 0]
```

In the Environment option of the main menu, check Everything. Using the option File/New Project, create a project. From the task menu, go to Project/Project Options, and check Show Constructors. Finally, go to the option Project/Update and Run. You are representing fractions using the Cartesian pair (p, q) , instead of

$$\frac{p}{q}$$

The set \mathbb{Q} is the set of rational numbers, i.e., numbers that can be represented as a fraction like

$$\frac{p}{q}$$

where $p \in \mathbb{Q}$ and $q \in \mathbb{Q}$.

The article that you are reading was used as a term paper. Therefore I needed to explain that there exist irrational numbers, i. e., numbers that one cannot write as fraction. A handy example of such a number is $\sqrt{2}$ (square root of 2), that is the result of dividing the side of a square by its diagonal. In the next page, I will prove to the satisfaction of teachers and parents that $\sqrt{2}$ is indeed an irrational number because one cannot write it as a fraction of two integers. I will use Latin expressions and talk about Greeks and Renaissance Italians. You will be better off skipping that stuff.

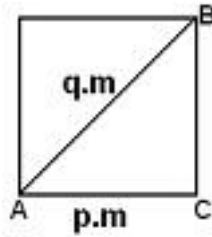


Figure 1: Simultaneous measurements

Skip from here: In Pythagora's time, Ancient Greeks claimed that any pair of line segments is commensurable, i.e., you can always find a meter, such that the lengths of any two segments are given by integers. An example will show you the Ancient Greek theory of commensurable lengths at work. Consider the square of figure 1. If the Greeks were right, I can find a meter, possibly a very small one, that produces an integer measure for the diagonal of the square, and another integer measure for the side. Suppose that p is the result of measuring the side of the square, and q is the result of measuring the diagonal. The Pythagorean theorem states that $\overline{AC}^2 + \overline{CB}^2 = \overline{AB}^2$, i.e.,

$$p^2 + p^2 = q^2 \therefore 2p^2 = q^2 \tag{1}$$

You can also choose the meter so that p and q have no common factors. For instance, if both p and q were divisible by 2, you could double the length of the meter, getting values no longer divisible by 2. E.g. if $p = 20$ and $q = 18$, and you double the length of the meter, you get $p = 10$, and $q = 9$. Thus let us assume that one has chosen a meter so that p and q are not simultaneously even. But from equation 1, one has that q^2 is even. But if q^2 is even, q is even too. You can check that the square of an odd number is always an odd number. Since q is even, you can substitute $2 \times n$ for it in equation 1.

$$2 \times p^2 = q^2 = (2 \times n)^2 = 4 \times n^2 \therefore \times p^2 = 4 \times n^2 \therefore p^2 = 2 \times n^2 \tag{2}$$

Equation 1 shows that q is even; equation 2 proves that p is even too. But this is against our assumption that p and q are not both even. Therefore, p and q cannot be integers in equation 1, which you can rewrite as

$$\frac{p}{q} = \sqrt{2}$$

Resume reading: Thus the number $\sqrt{2}$, that gives the ratio between the side and the diagonal of any square, is not an element of \mathbb{Q} , or else, $\sqrt{2} \notin \mathbb{Q}$. It was Hypasus of Metapontum, a Greek philosopher, who proved this. The Greeks, *un peuple de savants* (a people of wise men and women), had also the strange habit of consulting with an illiterate peasant girl at Delphi, before doing anything useful. Keeping with this tradition, Hyppasus asked the Delphian priestess— that illiterate girl— what he should do to please Apollo. The priestess told him to measure the side and the diagonal of the face of the god's cubic altar using the same meter. By proving that the problem was impossible, Hypasus discovered a type of number that can not be written as fraction. This kind of number is called irrational. A irrational number is not a crazy, or a stupid number; it is simply a number that you cannot represent as fraction, i.e., *ratio* in Latin, and *logos* in Greek.

The set of all numbers, integer, irrational, and rational is called \mathbb{R} , or the set of real numbers. In Clean, \mathbb{Z} is called **Int**, although the set **Int** does not cover all integers, but enough of them to satisfy your needs. A Clean real number belongs to the set **Real**, a subset of \mathbb{R} .

If $x \in \mathbf{Int}$, Clean programmers say that x has type **Int**. They also say that r has type **Real** if $r \in \mathbf{Real}$. There are other types besides **Int**, and **Real**. Here is a list of primitive types.

Int — Integer numbers between -2147483648 and 2147483647 .

Real — Reals must be written with a decimal point: 3.4 , 3.1416 , etc.

String — A quoted string of characters: `"3.12"`, `"Hippasus"`, `"pen"`, etc.

Char — Characters between single quotes: `'A'`, `'b'`, `'3'`, `' '`, `'.'`, etc.

2 Functions

A function is a relationship between an argument and a unique value. Let the argument be $x \in B$, where B is a set; then B is called domain of the function. Let the value be $f(x) \in C$, where C is also a set; then C is the range of the function. Functions can be represented by tables, plots and rules. Let us examine each one of these representations in turn.

2.1 Tables

Let us consider a function that associates **True** or **False** to the letters of the Roman alphabet. If a letter is a vowel, then the value will be **True**; otherwise, it will be **False**. The range of such a function is $\{\mathbf{True}, \mathbf{False}\}$, and the domain is $\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$.

Domain	Range	Domain	Range
a	True	n	False
b	False	o	True
c	False	p	False
d	False	q	False
e	True	r	False
f	False	s	False
g	False	t	False
h	False	u	True
i	True	v	False
j	False	w	False
k	False	x	False
l	False	y	False
m	False	z	False

2.2 Rules

From last section, you certainly perceived that it is pretty tough to represent a function using a table. You must list every case. There are also functions, like $\sin(x)$, whose domain has an infinite number of values, which makes impossible to list all entries. Even if you would try to insert only a finite subset of the domain into the table, it wouldn't be easy. Even so, in the past, people used to build tables. In fact, tables were the only way to calculate many useful functions, like $\sin(x)$, $\log(x)$, $\cos(x)$, etc. In 1814 Barlow published his Tables which give factors, squares, cubes, square roots, reciprocals and hyperbolic logs of all numbers from 1 to 10000. In 1631 Briggs published tables of sin functions to 15 places and tan and sec functions to 10 places. I heard the story of a mathematician who published a table of sinus, and made a mistake. Worried about hundred of sailors who lost their way due to his mistake, that mathematician committed suicide.

In order to understand how to use rules to represent a function, let us revisit the vowel table. Using rules, that table becomes

```
vowel x
| (x== 'a') || (x== 'e') || (x== 'i') || (x== 'o') || (x== 'u') = True
| otherwise = False
```

Functions has a parameter, also called variable, that represents an element of the domain. Thus, the vowel function has a parameter x , that represents an element of the set $\{'a', 'b', 'c', 'd', 'e', 'f', \dots\}$. Below the name of the function, and its variable, one finds a set of rules. Each rule has a condition between the signs $|$ and $=$. After the equal sign, one finds an expression that gives the value, if that rule applies. Consider the first rule:

```
| (x== 'a') || (x== 'e') || (x== 'i') || (x== 'o') || (x== 'u') = True
```

The condition is the boolean expression

```
(x== 'a') || (x== 'e') || (x== 'i') || (x== 'o') || (x== 'u')
```

The operator $||$ means **or**. Therefore, the boolean expression says that you can use the present rule if $(x== 'a')$, or $(x== 'e')$, or $(x== 'i')$, etc. In this case, the rule states that the functional value is **True**; the functional value is at the right hand side of the rule. Be careful not to mixup a double equal sign, that is an *equality* sign, and the single equal symbol, that introduces the functional value.

Now, let us consider the Fibonacci function, that has such an important role in the book "The Da Vinci Code". Here is its table for the first 7 entries:

0		1
1		1
2		2
3		3
4		5
5		8
6		13

Note that a function value is equal the sum of the two precedent values, or else, $\text{fib } n = \text{fib}(n - 1) + \text{fib}(n - 2)$. Assume that $n = 6$. Then,

$$\text{fib } 6 = \text{fib}(6 - 1) + \text{fib}(6 - 2) = \text{fib } 5 + \text{fib } 4$$

Of course, this rule is true only for $n > 1$, because there is not two precedent values for $n = 0$, or $n = 1$. An example shows how do you indicate that a rule is valid only under certain conditions.

```
module fibo
import StdEnv, myIO

Start world= ioDialog (ItoI fib) world

fib n
  |n > 1 = fib(n-1)+fib(n-2)
  |otherwise= 1
```

The function `ioDialog` will produce a graphic user interface for you. It has two arguments: `(ItoI fib)` and `world`. The first argument is the function `fib`; however, you must use the constructor `(ItoI fib)` to tell the system that `fib` is a function that take an integer value to another integer value, or else, the domain of `fib` is `Int`, and the range is also `Int`.

Clean has an *ObjectIO* library, that provides things like windows, and dialogs. In order to use it, you need to pass around a value of type `*World`. The operational system will call the `Start` function with a `world` value; thus you must feed the argument of `Start` to `ioDialog`:

```
Start world= ioDialog (ItoI fib) world
```

The steps to compile the `fibo` module are given below.

- From `File/New File`, create a file `fibo.icl`, with extension `icl`.
- In the `Environment` option of the main menu, check `Everything`.
- Create a project file using option `File/New Project`.
- Check `No Console` in `Project/Project Options`.
- Type in module `fibo`, given in the preceding page.
- `Project/Update and Run`.

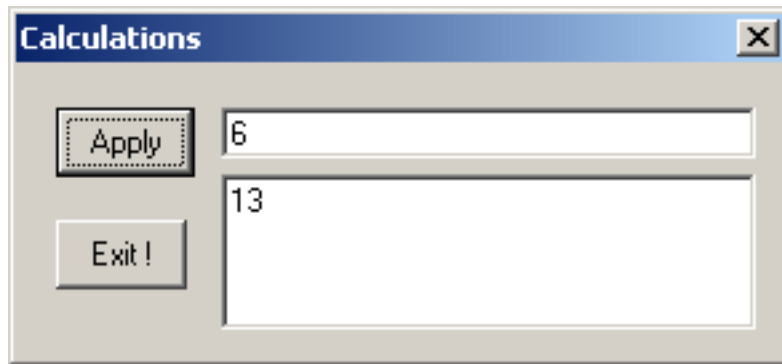


Figure 2: Fibonacci function

Figure 2 shows a session with the program that calculates Fibonacci numbers. To compile and use it, you must download and install the library `myIO`, provided in this site.

3 Factorial function

The factorial of n is the product of all integers between 1 and n . For instance, the factorial of 5 is given by

$$1 \times 2 \times 3 \times 4 \times 5$$

Let us build a table for the first seven entries of the factorial function.

0	1
1	1
2	2
3	6
4	24
5	120
6	720

If you take a careful look at this table, you will note that it has an interesting property: The factorial of n is equal to $n \times \text{factorial}(n - 1)$. Then, the factorial of 6 is equal to $6 \times \text{factorial}(5)$. This property is used in the rule based definition of the factorial function.

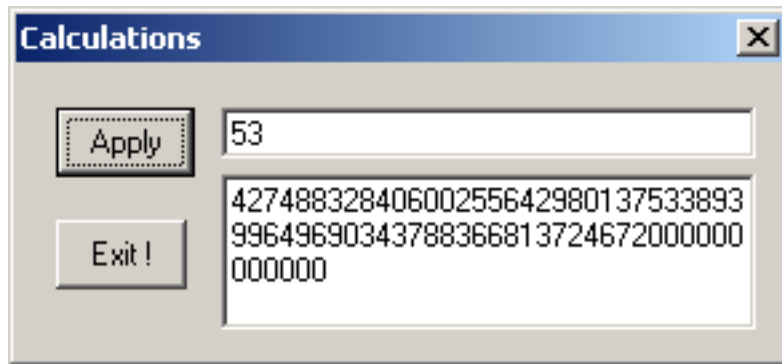


Figure 3: factorial function

```
module fact
import StdEnv, myIO

Start world= ioDialog (ZtoZ factorial) world

factorial::Z -> Z
factorial n
  | n==ZERO = ONE
  | otherwise= n*factorial(n-ONE)
```

The domain of `factorial` is `Z`; although type `Z` does not encompass all integers, it is based on Clean Extended arithmetic library, which is a pretty good approximation. Figure 3 shows factorial of 53.