Mathematical foundations

Table of Contents

1 Logic

Logic is a fundamental idea in both hardware and software. Hardware is made of two-state electronic switches which we think of as true or false. Programming languages have logical operators.

A *proposition* is a statement which is true or false. 'Paris is the capital of France' is a true proposition. '2+2=5' is a false proposition. 'Be quiet' is not a proposition at all (it is an imperative - an instruction).

A proposition has a *truth value*, which is either true or false. This data type, with just two values, true or false, is usually called a *boolean*, after George Boole, who developed many of these ideas.

There are three basic *logical operators*, 'not, 'and' and 'or'.

'*not*' inverts a truth value. So if p is true, not p is false. If p is false, not p is true. We can set this out in a *truth table*:

p	not p
false	true
true	false

In many programming languages, not is written !. So for example

```
if x!=5.. do something
```

means if x is not equal to 5, do something.

'and' works on two truth values. It means 'both':

p	q	p and q
false	false	false
false	true	false
true	false	false
true	true	true

'and' is written && in many languages. So

```
if x==3 && y==4.. do something
```

means if x is 3 and y is 4, do something

'or' means either or both:

p	q	p or q
false	false	false
false	true	true
true	false	true
true	true	true

'or' is often written as ||

These logical operators correspond exactly to logic gates in digital electronics

Test

1. What is a proposition?

2. What is the name of the data type with possible values of true and false only?

3. What is the truth value of

7 > 15 OR 2 < 3 ?

2 Sets

A mathematical set is simply a collection of things. For example {Paris, New York, Moscow} is a set of three cities. {8, 4} is a set of two integers.

Sets are the basic mathematical object. All other types of object - number, function, vectors, groups, fields, rings - are defined in terms of sets.

Sets are not ordered, so {3,2,1} is the same set as {1,3,2}

Sets have no duplicates. {2,2,2,3,3,4} is the same set as {2,3,4}.

The things in a set are called its *elements*. x ∈ S means 'x is an element of S'

We can combine sets in two ways - union and intersection, corresponding to the logical or and and.

The *union* of two sets is the things in either, *or* both, sets. So the union of {1,2,3} and {2,3,4} is {1,2,3,4}. This is written as ∪, so the union of sets A and B is A ∪ B.

The *intersection* is the elements in one *and* the other. So the union of {1,2,3} and {2,3,4} is {2,3}. This is written A ∩ B

A subset of a set is some (maybe all) the elements. So a subset of {1,2,3,4} is {1,3}.

Sets relate to some data structures.

Test

If set A is {5,6,8,9} and B is {9,6,4,2}, then

1. What is A ∪ B?

2. What is A ∩ B?

3. Is 6 ∈ A true or false?

4. Is {4,9} a subset of B?

3 The standard number sets

There are several different kinds of numbers

The *natural numbers* are 1,2,3,4.. Some people count 0 as a natural number. It is possibly to define the natural numbers in set terms (and the other kinds of numbers in terms of the natural numbers). Natural numbers are usually used for *counting things*. The set of natural numbers is usually written ℕ.

The *integers* are signed : .. -3, -2, -1, 0, +1, +2, +3.. The set of integers is ℤ (from the German Zahlen - numbers). Most programming languages have a data type called int, or similar. This is like ℤ, but is not infinite.

The *rational numbers* are fractions - like 1/4 and 3/4. There are also rationals greater than 1, like 4/3 and 27/6, and negative rationals like -2/3 and -14/3. On paper we can also write rationals as decimal expansions, such as 0.25 or 5.6713. Rational numbers can always be expressed as a/b where a and b are integers. The rational are ℚ, for quotient.

The *irrational numbers* cannot be written as a/b. An example is pi, which is about 3.14159265359.., and the square root of 2, which is around 1.41421356237.

The *real numbers* are all of these : integers, rationals and irrationals. They are called 'real' because there are also imaginary numbers, like the square root of -1. Measurements of quantities in science and engineering are usually real numbers. The reals are ℝ. Most programming languages have a data type of floating point numbers, which are like ℝ, but are not infinite, and have arithmetic of limited accuracy.

Ordinal numbers are used to label things, based on their position in some way. They are in contrast with cardinal numbers, which are used for *counting how many*. For example, a bus might carry the number 17. This does not mean there are 17 buses. It is a label, to show it is the bus on route number 17. The index of an array is in effect an ordinal number.

Test

1. Explain the difference between natural numbers and integers.

2. What is ℚ ?

3. In a programming language you know, what data type matches ℤ?

4 Functions

A function pairs up elements of one set with elements of another set.

For example one set might be {London, Paris, New York} and another set {red, green, blue}. One way to pair these is London - red, Paris - green, New York - blue. This is a mapping. Given one (say Paris), that maps to another (green).

This makes no 'sense'. We could have a different mapping - say London - red, Paris - green, New York - blue. That makes no sense either. Check London and New York both map to blue - no problem.

The only rule is that one element cannot map to two different ones. We cannot have London - red, London - green, New York - blue. This is simply how functions are defined - not 'one-to-many'.

We can also pair up elements of a set to elements of the same set. For example with the set {1,2,3}, we can have pairs (1,3), (2,2) and (3,1)

We often first meet functions like y(x) = 2x+1. This is a function from ℝ to ℝ. It pairs up 2 with 5, because 2 X 2+1=5. 3 pairs with 7, 4 with 9, and so on. This function has a formula : 2x+1. But a function does not *have* to have a formula. It is simply a set of pairs.

In software, a function might be called a *map*. A map is a set of key-value pairs. We can ask the map to fetch a value which has been paired with some key.

Test

If set A is {4,5,6} and B is {8,9,10}, then

1. Is the set of pairs (4,10) (5,9) (6,8) a function A → B?

2. (4,8) (4,9) (5,10) is not a function. Why not?

3. Is (4,8) (5,8) (6,8) a function?

5 Polynomials

A polynomial is a type of function ℝ→ℝ. For example

y(x) =$4x^3+5x^2+2x+4$ (this is a cubic - power of 3)

Another is

y(x) =$2x^2+3$ (this is a quadratic - power of 2)

and

y(x) =4x-1 (this is linear)

So a polynomial is the sum of some terms, each of which is a power of x, times a constant (called the coefficient).

Test

1. If y(x)=5x-2, what is y(3)?

2. What shape is the graph of a linear function?

3. Is y=e^x a polynomial?

6 Logarithms

A logarithm is another type of function. It is often abbreviated to 'log'.

Some examples:

$\log_{10} 100 = 2$

$\log_{10} 1000 = 3$

$\log_{10} 1000 = 4$

$\log_{10} 10= 1$

$\log_{10} 1/10 = -1$

So the log of a number is *what power we need to raise 10 to to get the number*.

In symbols $10^{\log x} = x$

This function is the *log to base 10*.

In computer science we more often use the log to base 2:

$\log_2 4 = 2$

$\log_2 8 = 3$

$\log_2 16 = 4$

We can think of the log to base 2 as the number of times we can divide by 2 (before getting down to 1). For example $\log_2 16 = 4$ because starting at 16 and dividing by 2 we get 8, 4, 2, and 1.

This is often used in Computer Science because many algorithms work by splitting a data set into two parts. The number of times we can do this is the log to base 2.

Test

1. What is the log to base 2 of 64?

2. 16 X 16 = 256. What is the log to base 2 of 256?