Lecture Notes: Quantitative Reasoning and Mathematical Thinking[1]

Subhashis Banerjee

Department of Computer Science
Ashoka University
Sonipat, Haryana, 131029
email: suban@ashoka.edu.in

September 3, 2025

Contents

Chapter 1

Introduction

Courant and Robbins, in What is Mathematics? (1941), present mathematics not as a dry collection of formulas and tools, but as a living, creative discipline rooted in human thought and curiosity. For them, mathematics is both a pathway for understanding the natural world and an autonomous intellectual pursuit that reveals structures of order, beauty, and generality. They stress that its essence lies in the interplay between abstraction and concrete problem-solving: starting from simple, practical problems, mathematics ascends to general concepts and theories that then illuminate new domains.

They emphasize accessibility and unity: mathematics belongs to everyone who is willing to think rigorously, and its spirit combines logic with imagination. Rather than reducing it to calculation or technical skill, Courant and Robbins describe mathematics as "an expression of the human mind" where precision, creativity, and aesthetic appreciation converge. Their central idea is that mathematics is at once useful, philosophical, and artistic—simultaneously a language of science, a training ground for reasoning, and a source of intellectual delight.

Early mathematics was computational when the emphasis was on finding methods to obtain solutions. However, over the years, the disciplines of mathematics and computer science – the subject of designing algorithms for problem solving – have diverged. In mathematics abstraction is symbolic and logical. It seeks general structures, patterns, and proofs independent of implementation. It often endeavours to seek and capture common structures across different abstractions. The primary aim is truth and understanding – developing rigorous proofs, ensuring logical consistency, and uncovering general laws. Utility often follows from this pursuit but is not always the main driver. In contrast, the role of abstraction in computational thinking is more operational and algorithmic. It emphasizes creating computational process models for natural, social and even abstract phenomena for operational analysis. The primary aim is effective procedure – designing algorithms that solve problems efficiently, often under constraints of time, memory, and real-world complexity. The power lies in execution and exploration—running a program can reveal insights about systems too complex to solve analytically. Both have become fundamental strands of epistemology that are essential for critical scientific thinking.

Data-driven inference represents a third way of knowing, distinct from the deductive rigour of mathematics and the constructive procedures of computational thinking. As practiced in modern data science and machine learning, it seeks knowledge not by proving theorems or designing explicit algorithms, but by discovering patterns and regularities directly from empirical data. Its epistemic core is induction at scale: hypotheses, models, or predictors are justified by their ability to capture hidden correlations and to generalize to new observations. Unlike mathematics, correctness is not absolute, and unlike computational thinking, procedures are not always fully transparent. Instead, credibility arises from empirical adequacy—the degree to which models explain, predict, or align with observed phenomena. This mode of inference expands our epistemic toolkit for a world where complexity and abundance of data overwhelm deductive or constructive methods, but it also brings new philosophical challenges: uncertainty about correctness, bias, and the gap between correlation and causation.

In this course we will try to cover some fundamentals of all of the above.

Chapter 2

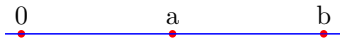
God gave us numbers, and human thought created algorithms

2.1 Numbers

Our discussion on mathematics and computing must start with numbers. What are numbers after all? The same number may be represented with symbols such as 3, III, or even as a line of a fixed length. But what is the underlying concept behind the different representations?

Bertrand Russell defined numbers as sizes (or *cardinality*) of *collections*. Some examples of *equinumerous* collections are $\{Red, Blue, Green\}$, $\{Amir, Salman, Shahrukh\}$, $\{Godavari, Kaveri, Krishna\}$. *Collections* are also called *Sets* or *Classes* in Mathematics. All three Sets above are of cardinality 3. Russel defined a number as – *the number of a class is the class of all those classes that are similar (equinumerous) to it.* So, according to Russel, a number is a class of classes.

Note that *cardinality* of sets is not the only way to describe the concept of a number. A number may also be a measure of a length. For example, in the straight line below, if we define the segment $\overline{0a}$ to be the *unit length* representing the number 1, then the line segment $\overline{0b}$ which is twice the length of $\overline{0a}$ may represent the number 2.



Without belabouring the point, it will suffice to say for our purpose that all of us intuitively understand what numbers mean.

2.1.1 Numbers may be represented in multiple ways

However, we need to do useful stuff with numbers – we need to add, subtract, multiply and divide them for obvious practical reasons. Indeed, the history of numbers date back to the Mesolithic stone age. The early humans had to figure out – due to a variety of practical considerations – that if they put two similar collections of size two and size three together, the larger collection becomes of size five.

Civilisations have found many ways to represent numbers through the ages. Some examples are as tally marks in the prehistoric to early civilisations – as straight marks on bones, sticks, or stones – as can be observed in the archaeological evidence of the Ishango bones from around 20000 BCE; as Egyptian – a stroke for 1, heel bone for 10, coil of rope for 100, etc. – or Roman – I, V, X, L, C, D, M – numerals; as Base-60 (sexagesimal) numbers written as combinations of "1" and "10" wedges

by the Babylonians around 2000 BCE; as used rods arranged on counting boards in base-10 with positional notation in Chinese rod systems; as positional decimal systems in Indian numerals in the Gupta period around 5^{th} century CE; as beads or stones moved on rods or grooves to represent numbers in Abacus systems in China, Rome, Mesopotamia and Jerusalem; with Indo-Arabic numerals in the medieval period; with various mechanical calculators such as Napier's bones, Slide rules, Pascal's calculator, and Leibniz's stepped reckoner in the 17^{th} century; as gears and levers in Charles Babbage's first programmable computer – the Analytic Engine; and as bits and bytes in modern digital computers. Note, also, that the methods of carrying out these operations – the algorithms – will necessarily depend on the representation we choose for numbers.

2.2 Sets

We will use *Sets* quite a bit in this course. We may describe a *Set* or a *Collection* by explicitly listing out its elements without duplicates, such as in the examples above. We may sometimes also describe a Set with a property like "all students enrolled in the QRMT section FC-0306-3". We write this formally using a variable x as $\{x \mid x \text{ is a student in the QRMT section FC-0306-3}\}$. The symbol $\mid$ is read as "such that".

If an element x belongs to a set A, we usually write this as $x \in A$.

Here are some more examples of Sets:

1. $A = \{x \mid x \text{ is a student pursuing a degree in India}\}$
2. $B = \{x \mid x \text{ is a CS Major student at Ashoka University}\}$
3. $C = \{x \mid x \text{ is a CS Major student at Ashoka University } \textbf{and } x \text{ is female}\}$

Clearly, all members of C are also members of B, and all members of B are members of A. We then say that C is a *subset* of B ($C \subseteq B$), and B is a *subset* of A ($B \subseteq A$). Formally, a set B is a *subset* of another set A, denoted as $B \subseteq A$, if $x \in A$ whenever $x \in B$. The empty set is denoted by ϕ, its size is zero (0), and it is a subset of all sets.

Given two sets A and B, the *union* $A \cup B$ is the set of all elements that are in A, or in B, or in both. Formally, $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$. For example, if $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$, then $A \cup B = \{1, 2, 3, 4, 5\}$.

Given two sets A and B, the *intersection* $A \cap B$ is the set of all elements that are in both A and B. Formally, $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$. For example, if $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$, then $A \cap B = \{3\}$.

Clearly, for any set A, $A \cup \phi = A$ and $A \cap \phi = \phi$,

Exercise 2.1 *Suppose $B \subseteq A$. Argue that*

1. $A \cup B = A$
2. $A \cap B = B$

2.3 The set of Natural numbers

Some sets can also be unbounded or infinite. We define the set of *Natural numbers* as $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$[1].

While we all intuitively understand this set, note that the elements of the set are as yet uninterpreted and undefined. We can overcome this lacunae by assuming a *God-gifted* ability to count. Given a number n as the size of a Set or a length, let us assume that we can interpret and construct the successor of n as $S(n) = n + 1$. Then, we can formally define the set of Natural numbers $\mathbb{N}$ as

1. $0 \in \mathbb{N}$, where 0 is the symbol that denotes the size of the empty set, and

[1] 0 is usually not included in the Set of Natural numbers in Mathematics. We will however include 0 in the set of Natural numbers in this course. After all, it is quite natural to score a 0 in an examination

2. if $n \in \mathbb{N}$, then $S(n) = n + 1 \in \mathbb{N}$

We can then adopt a suitable representation for successive elements in the set $\mathbb{N}$. Note that the set $\mathbb{N}$ is unbounded, because every number – no matter how large – has a successor.

2.3.1 Addition

We observed that the underlying concept of a number is independent of specific representations. Ideally, so should be the concepts of carrying out various operations with numbers. We may think of addition – the sum $a + b$ of two numbers a and b – as just combining two similar sets of sizes a and b. However, the procedure for "combining" is not representation independent. While simple "putting together" may work if we represent the numbers as collections of stones or marbles, it is not well defined for adding two numbers in the place-value representation that we are familiar with from junior school. Hence "combining" is a somewhat unsatisfactory way of defining addition.

A better way of defining $a + b$ is by using the successor operation $S(a) = a + 1$, b times. As long as we have a primitive method for computing $a + 1$ in any representation for an arbitrary a, this definition of $a + b$ becomes representation independent. We may define the basic property of addition using counting as:

For all $m, n \in \mathbb{N}$:

1. $m + 0 = m$
2. $m + S(n) = S(m + n)$

In the above definition we have used the same trick as in definition of the set $\mathbb{N}$ above, of defining a larger concept as a successor of a smaller concept. The process repeats, and the actual additions happen in the return path. For example,

$$
\begin{aligned}
&7 + 5 \\
&\quad = (7 + 4) + 1 \\
&\quad = ((7 + 3) + 1) + 1 \\
&\quad = (((7 + 2) + 1) + 1) + 1 \\
&\quad = ((((7 + 1) + 1) + 1) + 1) + 1 \\
&\quad = (((((7 + 0) + 1) + 1) + 1) + 1) + 1 \\
&\quad = ((((7 + 1) + 1) + 1) + 1) + 1 \\
&\quad = (((8 + 1) + 1) + 1) + 1 \\
&\quad = ((9 + 1) + 1) + 1 \\
&\quad = (10 + 1) + 1 \\
&\quad = 11 + 1 \\
&\quad = 12
\end{aligned}
$$

Note that the repeated substitution of a larger problem with a smaller problem is bounded, because the first condition of the definition works as a sentinel that we are bound to encounter as we keep reducing n.

We can then describe a procedure for computing $a + b$ (Algorithm 1) based on the above principle, but avoiding the deferred computations. The procedure takes a and b as input and returns *sum* as the output. $sum \leftarrow sum + 1$ denotes the operation "*sum* is *assigned sum* $+ 1$" indicating that *sum* is incremented by 1.

Exercise 2.2 *1. Assuming that the operation $a + 1$ is available as a primitive, convince yourself that the above procedure for adding two numbers are correct.*

2. Argue that if the operation $a+1$ is available as a primitive, then the above algorithm for addition is representation independent.

3. Describe how the algorithm may be implemented using pebbles or marbles to represent numbers.

Algorithm 1 An algorithm for $a + b$ by $+1$ b-times.

```
1: procedure ADD(a, b)
2:     counter ← 0
3:     sum ← a
4:     while counter < b do
5:         sum ← sum + 1
6:         counter ← counter + 1
       return sum
```

2.3.2 Multiplication

We can now define multiplication as repeated additions:

1. $n \times 0 = 0$, for all $n \in \mathbb{N}$
2. $n \times S(m) = n \times m + n$, for all $n, m \in \mathbb{N}$

Note that here again we have defined $n \times S(m)$, in terms of a smaller problem $m \times n$ of the same type.

Exercise 2.3 *1. Convince yourself that according to the above definition* $n \times m = \underbrace{n + n + n + \ldots + n}_{m \text{ times}}$.

2. Provide a representation independent algorithm, using only the successor function and addition, for multiplication of two numbers.

3. Describe how the algorithm may be implemented using pebbles or marbles to represent numbers.

2.3.3 Subtraction

To define the subtraction operation $m - n$, we may first define a predecessor operation $P(n)$ – analogous to $S(n)$ – as

1. $P(0)$ is undefined
2. $P(n) = n - 1$ for all $n > 0$.

We assume, as before, that we have a primitive counting based procedure for computing $P(n) = n - 1$ in any representation. We can define the subtraction operation $m - n$ similarly to addition:
For all $m, n \in \mathbb{N}, m \geq n$

1. $m - m = 0$
2. $m - n = S(P(m) - n)$

As before, note that $P(m) - n$ is a smaller problem that $m - n$.
The subtraction algorithm may then be given as:

Algorithm 2 An algorithm for $a - b$, $a \geq b$ by -1 b-times.

```
1: procedure SUBTRACT(a, b)
2:     counter ← 0
3:     while counter < b do
4:         a ← a − 1
5:         counter ← counter + 1
       return a
```

Exercise 2.4 *Provide alternative versions of Algorithms 1 and 2 without using the counter. Instead decrement* b *using* $b \leftarrow b - 1$ *repeatedly till* $b = 0$.

2.3.4 Division

Division is a natural requirement in civilised societies, mainly for sharing. However, it may not always be possible to divide natural numbers in equal proportions. For example, a collection of size 3 cannot be divided in two proportions of equal sizes without breaking up at least one member element. We have the *division theorem*:

Theorem 2.1 *Given two numbers $a, b \in \mathbb{N}$, there exist unique $q, r \in \mathbb{N}$ (quotient and remainder, respectively) such that $a = bq + r$ and $0 \leq r < b$.*

Proof: Let us first argue that such q and r exist. Repeatedly compute $a-b, a-2b, a-3b, \ldots, a-kb$, $k \geq 0$, till $a - kb < b$ and subtraction is possible no more. Set $q = k$ and $r = a - kb$. Clearly, q is the total number of times b can be subtracted from a, and $0 \leq r < b$. If $r = 0$ then b divides a exactly.

To argue that that q and r obtained by the above procedure are unique, let us suppose they are not. Then, there exist q_1, r_1 and q_2, r_2 such that

$$a = bq_1 + r_1, 0 \leq r_1 < 0$$
$$a = bq_2 + r_2, 0 \leq r_2 < 0$$

Without loss of generality, let us assume that $q_1 \geq q2$. The above implies that $b(q_1 - q_2) = r_2 - r_1$. One of two cases arise:

1. $q_1 = q_2$. This implies that $r_1 = r_2$, and hence uniqueness.
2. $q_1 > q_2$. This implies that $q_1 - q_2 \geq 1 \in \mathbb{N}$. Hence $r_2 - r_1 \geq b$. But this is not possible because $0 \leq r_1, r_2 < b$. Hence q and r must be unique.

□

In the above proof, we used *explicit construction* as a proof technique for establishing existence of such q and r, and *contradiction* for establishing their uniqueness. We will revisit these techniques later in the course when we discuss proofs.

Exercise 2.5 *Describe an algorithm using repeated subtraction that computes q and r given a and b.*

2.4 The Sets of Integers

We defined the subtraction operation $m - n, m \geq n, m, n \in \mathbb{N}$ as the number of times the successor operation $S()$ needs to be applied to reach m from n. This definition requires the restriction that $m \geq n$. An obvious generalisation is to remove the restriction and measure the difference in terms of either the successor $S()$ or the predecessor $P()$ operator. Subtraction then becomes directional, and we require negative numbers to represent the direction. This leads us to the set of integers

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, \ldots\}$$

Arithmetic in the set $\mathbb{Z}$ follows the same principles as in $\mathbb{N}$, except that they are now directional.

Exercise 2.6 *Rework the definitions and the algorithms for addition, multiplication, subtraction and division in $\mathbb{Z}$.*

2.5 The Sets of Rationals

The division theorem tells us that given $m, n \in \mathbb{N}$, there exist $q, r \in \mathbb{N}$, such that m can be divided in to q parts of size n, possibly leaving a remainder $0 \leq r < n$. Division is an obvious fundamental need for resource sharing. If each unit is indivisible – like live cattle, for example – then the division theorem is the best we can do. However, items measured in units such as weight, volume or length – such as meat from a hunted animal, or a pile of grains – are often divisible in smaller proportions

like $1/3^{rd}$, $2/25^{th}$ etc. So, division inevitably leads us to fractions. We define the set of Rational numbers as

$$\mathbb{Q} = \{x | x = p/q, p \in \mathbb{Z}, q \in \mathbb{N}, q \neq 0\}$$

We often also write this as $\frac{p}{q}$. These are numbers of the type $\pm 1/1, \pm 1/2, \pm 1/3, \pm 1/4, \pm 2/5$ etc. We may also insist that p and q should have no common factors (i.e., $gcd(p, q) = 1$; see Section 3.5 for a formal definition of *gcd*) to avoid multiple representations for the same Rational number. Clearly $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q}$.

Note, however, that we now have a situation where between any two rational numbers there are infinitely many other rational numbers.

Exercise 2.7 *Convince yourself of the above statement.*

This implies that there is no well defined successor function for a rational number, and we need to revisit our definition of addition for rationals. We start by noting that, for example,

$$\frac{1}{3} + \frac{2}{3} = \frac{1+2}{3} = 1$$

i.e., we can add the numerators as in $\mathbb{Z}$ if the denominators are the same. However, the addition

$$\frac{2}{3} + \frac{3}{4}$$

is not well defined unless the two fractions can be expressed in the same unit. But we can multiply the numerator and denominator of the first fraction by 4, and the second by three to convert to the same unit where the denominator of both is 12

$$\frac{2 \times 4}{3 \times 4} + \frac{3 \times 3}{4 \times 3} = \frac{8}{12} + \frac{9}{12} = \frac{8+9}{12} = \frac{17}{12}$$

Note that multiplying the numerator and the denominator of a fraction with the same number does not change the fraction. So, we can define the general rule for addition of two rational numbers as

$$\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 \times q_2 + p_2 \times q_1}{q_1 \times q_2}$$

where all the additions and multiplications are defined on the set $\mathbb{Z}$.

Exercise 2.8 *Extend the above idea to define subtraction, multiplication and division in the set* $\mathbb{Q}$.

Chapter 3

Ruler and compass algorithms

Figure 3.1: Ruler and compass. We will assume that the ruler is unmarked, and lengths can only be measured by adjusting the width of the compass.

Our endeavour so far has been to define numbers, and operations on them, in a representation independent manner. Let us now consider a specific computational model – straightedge and compass constructions introduced by the ancient Greeks – and examine whether the abstract operations we have defined above can be translated in to definite constructible procedures, or *algorithms*. Most of the geometric constructions date back to Euclid's books of Elements from around 300 BCE. We will often – by force of habit – refer to them as *ruler and compass constructions* but with the understanding that the ruler has no markings for length measurements, and can only be used to draw straight edges. See Figure 3.1.

Let us first consider some basic constructions. In what follows, the correctness of the constructions will often rely on the basic geometric properties of similar triangles. A reader may revise them from here and here.

3.1 Constructing a line perpendicular to a given line passing through a point

Given a straight line ℓ, and a point x on it, let us consider the problem of constructing a line perpendicular to ℓ and passing through x. We give a construction in Algorithm 3 (See Figure 3.2).

We have described the algorithmic procedure using some standard primitives. $c = CIRCLE(x, r)$ denotes the construction of a circle c centred at x of radius r. A and B are the intersection points of c with ℓ, denoted in the algorithm as $(A, B) = c \cap \ell$. Similarly, c_A and c_B are circles of radius $2r$

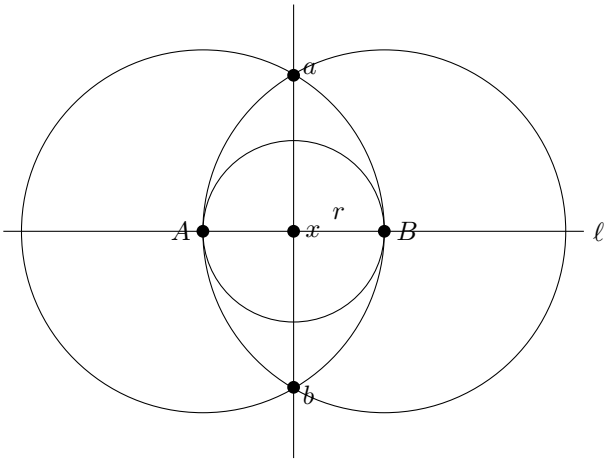


Figure 3.2: Constructing a line perpendicular to a given line passing through a point

Algorithm 3 Constructing a line perpendicular to a given line passing through a point

1: **procedure** PERPENDICULAR(x, ℓ)
2: $c = CIRCLE(x, r)$, where r is a random length
3: $(A, B) = c \cap \ell$
4: $c_A = CIRCLE(A, 2r)$
5: $c_B = CIRCLE(B, 2r)$
6: $(a, b) = c_A \cap c_B$
7: $result = LINE(a, b)$

centred at A and B respectively, and a and b are the intersection points of c_A and c_B. $LINE(a, b)$ joins a and b and is the *result*.

Exercise 3.1 *1. Convince yourself that the above construction is correct. Use properties of similar triangles.*

2. Argue that $LINE(a, b)$ is also the perpendicular bisector of AB.

3.2 Constructing a line parallel to a given line passing through a point

Given a line ℓ, and an arbitrary point x, consider the problem of construction of a line parallel to ℓ passing through x. We give a construction in Algorithm 4 (See Figure 3.3).

Note that in steps 4,6 and 7 of the algorithm, we have used the procedure of Algorithm 3. We will routinely use a previously defined algorithm as a primitive to define a new algorithm.

3.3 Constructibility and the compass equivalence theorem

The above two sections give us several examples of construction of points, lines, line segments and circles. The informal definition of *constructibility* is as follows. Given points are by definition constructible. A line joining two constructible points is constructible. So is the circle centred on one constructible point passing through another constructible point. A point is constructible if it is an intersection of constructible lines and circles.

The compass advocated by the Greek philosopher Plato in these constructions is a *collapsing compass*, i.e., a compass that "collapses" whenever it is lifted from a page, so that it may not be

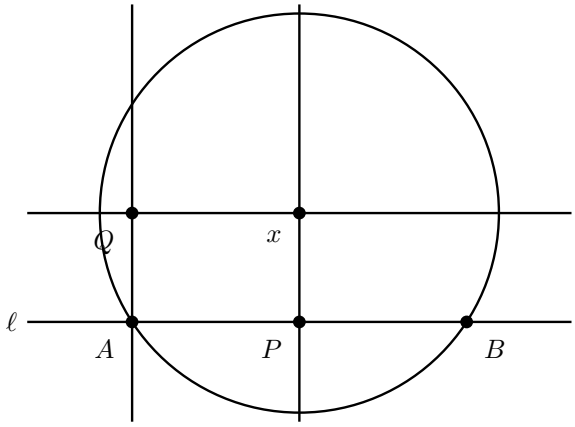


Figure 3.3: Constructing a line parallel to a given line passing through a point

Algorithm 4 Constructing a line parallel to a given line passing through a point

1: **procedure** PARALLEL(x, ℓ)
2: $c = CIRCLE(x, r)$, where r is a random length
3: $(A, B) = c \cap \ell$
4: Construct p, the perpendicular bisector of AB using Algorithm 3. Argue that p passes through x.
5: $P = p \cap \ell$
6: Construct q, a perpendicular to ℓ passing through A using Algorithm 3.
7: Construct s, a perpendicular to p passing through x using Algorithm 3.
8: $Q = q \cap s$
9: $result = LINE(Q, x)$

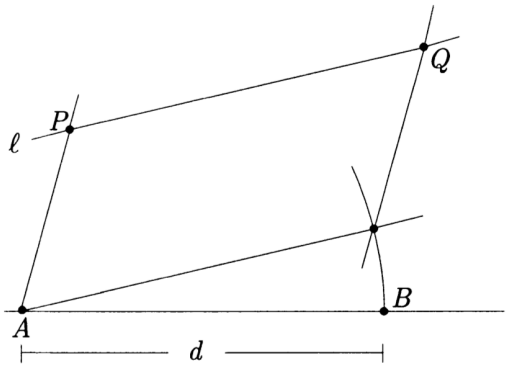


Figure 3.4: A compass-equivalence construction

directly used to transfer distances unlike in a modern fixable aperture compass. Note that nowhere in Sections 3.1 and 3.2 have we transferred distances using a fixed size compass lifted from the page.

This is however not a limitation as the following construction shows.

Theorem 3.1 *A collapsing compass can be used to transfer a given length to an arbitrary given point on a given line. That is, given an arbitrary length $\overline{AB}$, a line ℓ and a point P on it, it is possible to construct using a collapsing compass a point Q on ℓ such that $\overline{PQ} = \overline{AB}$.*

Proof:

Algorithm 5 A compass-equivalence construction

1: **procedure** COMPASS-EQUIVALENCE(A,B,ℓ,P)
2: $c = CIRCLE(A, B)$; note that $\overline{AB} = d$
3: Construct p, the line parallel to ℓ passing through A using Algorithm 4
4: $R = c \cap p$
5: $q = LINE(A, P)$
6: Construct s, the line parallel to q passing through R using Algorithm 4
7: $Q = s \cap \ell$
8: $\overline{PQ}$ is the result on ℓ

Exercise 3.2 *Convince yourself that $\overline{PQ} = \overline{AB}$.*

□

The above is different from the original construction and proof of correctness as given by Euclid in his Book of Elements. Interested readers may study a modern version of the original construction here.

3.4 Rational numbers are constructible

First, we can consider any given length as 1, measure it using a non-collapsing compass – which we just proved is equivalent to a collapsing compass – and add it to any point on a given line by marking it off with the compass. We can similarly subtract. The operations $S(n)$ and $P(n)$ are thus realisable using ruler and compass. Consequently, the elements of the set $\mathbb{Z}$ are constructible. In Figures 3.5 and 3.6 we give the direct constructions for $a + b$ and $a - b$.

Exercise 3.3 *1. Describe a ruler and compass procedure for multiplication using repeated additions.*

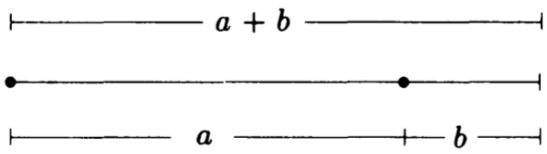


Figure 3.5: Construction of $a + b$.

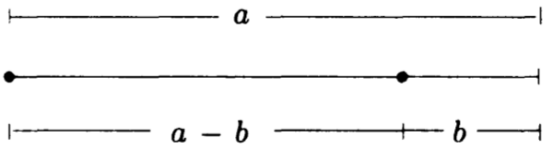


Figure 3.6: Construction of $a - b$.

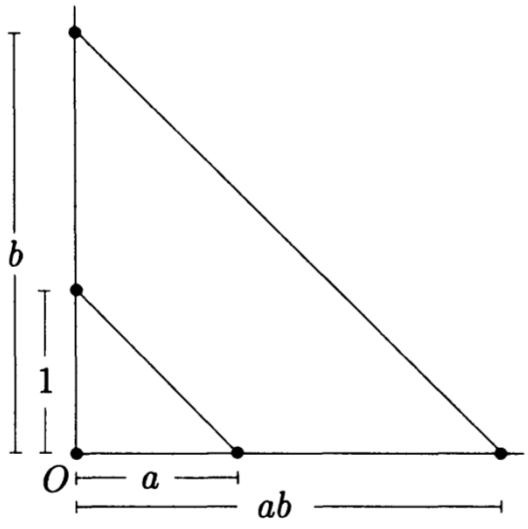


Figure 3.7: Construction of ab given a and b.

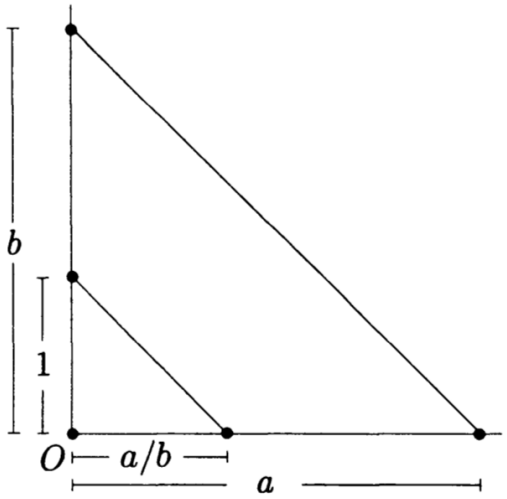


Figure 3.8: Construction of a/b given a and b.

2. *Describe a ruler and compass procedure for division (computing quotient and remainder) using repeated subtractions.*

Given integers a and b as line segments, we can also construct rational number ab and a/b directly using similar triangles. The construction of Figure 3.7 involves marking off the lengths a and b in two perpendicular segments from O, constructing the unit length in the direction of b, and constructing a line parallel to the line $\overline{a1}$ through a or b. The intercept of the parallel line in the direction of a then marks the length ab by similarity of the triangles.

We can similarly construct a/b as depicted in Figure 3.8. Rational numbers are thus constructible.

3.5 Euclid's GCD using ruler and compass

GCD of two integers $a > 0$, $b \geq 0$ is defined as the largest integer $d, d > 0$ that divides both a and b. Consider the following algorithm for computing the GCD:

$$gcd(a,b) = \begin{cases} a & \text{if } b = 0 \\ a & \text{if } a = b \\ gcd(a-b,b) & \text{if } a > b \\ gcd(a,b-a) & \text{if } b > a \end{cases}$$

Exercise 3.4 1. *Convince yourself that the above algorithmic specification (rule) is correct for computing GCD. Carry out the pencil and paper computation using the above algorithm for some special cases.*

2. *Describe the procedure for executing the algorithm using ruler and compass.*

The algorithm described above is from Euclid's Elements. You can find a description of it here. This is also considered to be the oldest non-trivial algorithm in common use.

Now that we have defined our first computational model, several questions arise. What are the full powers of the model? What are the other things that can be constructed? What are the limits of the model, and are there easily defied concepts that are not constructible? Can there be other computational models more powerful than ruler and compass? These are the kind of questions we interrogate every computational model with. We will revisit some of these questions in the latter chapters.

Chapter 4

Abstraction turns problems and concepts into principles

Modern mathematics and computer science are built upon a few simple but very powerful ideas. Among the most important are the notions of *relations* and *functions*, which allow us to describe how objects are connected or transformed. Counting, infinity, and the ways in which sets can be grouped into classes help us measure size, structure, and complexity. These ideas may look abstract at first, but they underlie the methods used in algorithms, data structures, and logical reasoning.

In computer science, functions capture the essence of computation: a program takes inputs and produces outputs, just as a function does. Understanding one-one and onto functions helps us reason about whether information is lost, preserved, or fully covered. Equivalence classes and partitions allow us to organise data into categories. Modular arithmetic, often called "clock arithmetic," is fundamental in cryptography, error detection, and digital systems. Learning these concepts gives us a foundation to explore deeper mathematics and to apply it to practical computational problems.

4.1 Relations

The *Cartesian product* of two sets A and B, denoted by $A \times B$, is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$. Thus,

$$A \times B = \{(a, b) \mid (a \in A) \text{ and } (b \in B)\}$$

A^n is the set of all ordered n-tuples $(a_1, a_2, \ldots, a_n)$ such that $a_i \in A$ for all i. i.e.,

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n \text{ times}}$$

A *relation* tells us which elements of one set are connected to elements of another.

A *binary relation* R from A to B is a subset of $A \times B$. It is a characterisation of the intuitive notion that some of the elements of A are related to some of the elements of B.

If $A = \{1, 2, 3\}$ and $B = \{a, b, c\}$, then $R = \{(1, a), (2, b), (3, a)\}$ is a relation from A to B. Familiar binary relations from $\mathbb{N}$ to $\mathbb{N}$ are $=$, $\neq$, $<$, $\leq$, $>$, $\geq$. Thus the elements of the set $\{(0, 0), (0, 1), (0, 2), \ldots, (1, 1), (1, 2), \ldots\}$ are all members of the relation $\leq$ which is a subset of $\mathbb{N} \times \mathbb{N}$.

4.2 Function

A *function* from A to B – written as $f : A \to B$ – is a special relation in which:

1. every element of A is related to some element of B, and

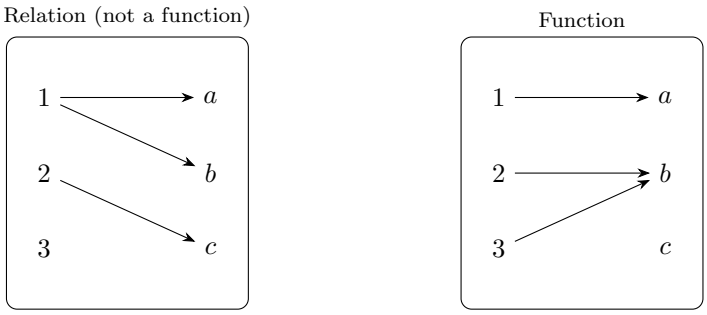


Figure 4.1: Relation vs function: In a relation, an element in A can be mapped to multiple elements in B, but not so in a function. In a function, all elements in A must be covered, but not necessarily so in a relation. The set B need not be fully covered in either.

2. no element of A is related to more than one element of B.

Equivalently, each input has *exactly one* output.

Some familiar examples of functions are

1. $+$ and $*$ (addition and multiplication) are functions of the type $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
2. $-$ (subtraction) is a function of the type $f : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$.
3. *div* and *mod* are functions of the type $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. If $a = q * b + r$ such that $0 \leq r < b$ and $a, b, q, r \in \mathbb{N}$ then the functions *div* and *mod* are defined as $div(a, b) = q$ and $mod(a, b) = r$. We will often write these binary functions as $a * b$, a *div* b, a *mod* b etc.
4. The binary relations $=$, $\neq$, $<$, $\leq$, $>$, $\geq$ are also functions of the type $f : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ where $\mathbb{B} = \{false, true\}$.
5. $f : \mathbb{N} \to \mathbb{N},\ f(x) = x^2$.

We write the definition of a function formally as follows.

A *function* from A to B is a binary relation f from A to B such that for *every* element $a \in A$ there is a *unique* element $b \in B$ so the $(a, b) \in f$ (or $f(a) = b$)[1]. We will use the notation $f : A \to B$ to denote a function f from A to B. The set A is called the *domain* of the function f and the set B is called the *co-domain* of the function f. The *range* of a function $f : A \to B$ is the set $\{b \in B \mid \text{for some } a \in A,\ f(a) = b\}$ denoting the subset of elements in B that are actually covered by f.

4.2.1 One-One (injective), Onto (surjective), and bijective Functions

When we study functions, it is often not enough to know that "every input has exactly one output." We also want to understand how well the function uses its codomain and whether different inputs remain distinct after applying the function.

Think of a classroom with students and seats:

- If no two students sit in the same seat, the "assignment" of students to seats is one-one (injective).
- If every seat is occupied by at least one student, the assignment is onto (surjective).
- If both conditions happen together — each student has exactly one seat, and every seat is filled — then the assignment is bijective. In such a case we may also define an inverse function from seats to students.

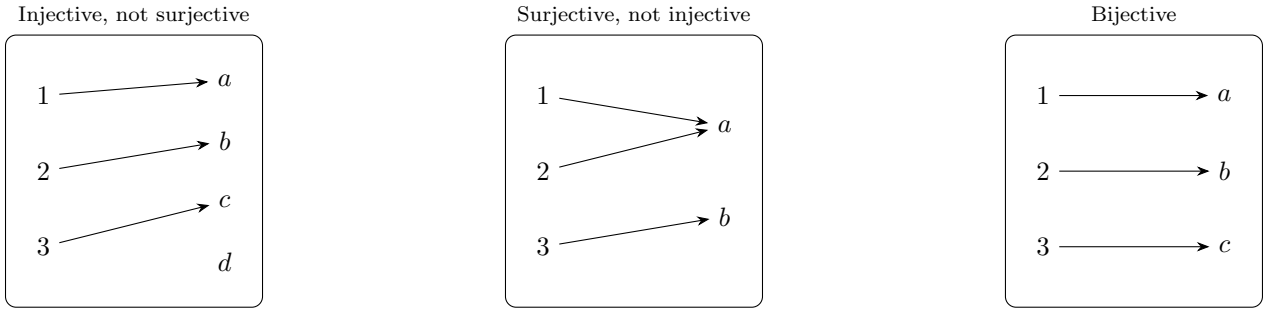


Figure 4.2: *One-one (injective)*: different inputs give different outputs. (like roll numbers: no two students share one roll); *Onto (surjective)*: every element of the codomain gets hit by *some* input. (no empty seats.); *Bijective*: both one-one and onto. (perfect pairing; has an inverse.)

The formal definitions below make the concepts precise.
Let $f : A \to B$ be a function. f is

Injective if whenever $f(a_1) = f(a_2)$ for $a_1, a_2 \in A$, we can conclude that $a_1 = a_2$

Surjective if for all $b \in B$, there exists $a \in A$ such that $f(a) = b$.

Bijective if it is both injective and surjective.

Examples.

- $f : \mathbb{N} \to \mathbb{R},\ f(n) = 2n$ is injective but not surjective (odd numbers are not covered).
- $g : \mathbb{N} \to \mathbb{N},\ g(0) = 0;\ g(n) = n - 1$ is surjective but not injective (why?).

[1]This is sometimes written using mathematical notation as $\forall a \in A$, $\exists$ unique $b \in B$. $\forall$ is the usual symbol for *for all*, and $\exists$ is the usual symbol for *there exists*