Abstract Algebra

Rohan Jain

Contents

Chapter 1

1.1 Introductory Notes

1.1.1 Things to Remember

Note:

- Definitions will usually be stated as "if" even though they mean "if and only if".
- Any form of proof is valid. Avoid proofs by contradiction because of disbelief in the law of excluded middle.
- When you define an object, you can *only* utilize its definition to prove anything about it.

1.1.2 Set Review

Definition 1.1.1: Set

In mathematics, a set is an undefined term. Basically, "everyone knows what it is." A few examples of sets are:

- The empty set is the set with no elements. It is denoted by ϕ or $\emptyset$.
- $\mathbb{N}$ is the set of natural numbers.
- $\mathbb{Z}$ is the set of integers.
- $\mathbb{Q}$ is the set of rational numbers.
- $\mathbb{R}$ is the set of real numbers.
- $\mathbb{C}$ is the set of complex numbers.

Note:

- A set is a well-defined collection of objects. The objects in a set are called elements of the set.
- A set is generally defined as a capital letter.
- $(A = B) \iff (\forall x : x \in A \iff x \in B)$
- $(A \subset B) \iff (\forall x \in A : x \in B)$
- A is a proper subset of B if $A \subset B$ and $A \neq B$.

Theorem 1.1.1

$A = B \iff A \subset B \wedge B \subset A$

Note:

- $A \cup B = x : x \in A \vee x \in B$
- $A \cap B = x : x \in A \wedge x \in B$
- $A \backslash B = x : x \in A \wedge x \notin B$
- $C \backslash (A \cup B) = (C \backslash A) \cap (C \backslash B)$

1.1.3 Cartesian Products and Functions

Note:

- $A \times B = \{(a, b) : a \in A \wedge b \in B\}$

Example 1.1.1 (Cartesian Product of two sets)

Let $A = \{1, 2, \Delta\}$ and $B = \{0, \pi\}$

- $(1, 0)$
- $(2, 0)$
- $(\Delta, 0)$
- $(1, \pi)$
- $(2, \pi)$
- (Δ, π)

Note:

Relations are subsets of Cartesian Products. For example, we can say that $<$ is a relation on the subset of $\mathbb{R} \times \mathbb{R}$ consisting of all ordered pairs of real numbers such that the first element is less than the second.

Definition 1.1.2: Function

A function f from a set A to a set B is a subset of $A \times B$ such that for every $a \in A$, there is exactly one $b \in B$ such that $(a, b) \in f$.

Note:

Let R be a relation from A to B.

- A is the domain
- B is the codomain
- $\{b : aRb\}$ is the image
- R is injective (one-to-one) if $a_1Rb \wedge a_2Rb \implies a_1 = a_2$
- R is surjective (onto) if $\forall b \in B : \exists a \in A : aRb$. Basically if the image is the entire codomain.
- R is bijective if it is injective and surjective

Note:

$A \xrightarrow{\text{R}} B$
$B \xrightarrow{\text{S}} C$
Define the composition as $S \circ R = \{(a, c) : \text{there is some } b \text{ such that } (a, b) \in R \text{ and } (b, c) \in S\}$

Theorem 1.1.2

Let $f : A \longrightarrow B$, $g : B \longrightarrow C$, and $h : C \longrightarrow D$. Then

- $h \circ (g \circ f) = (h \circ g) \circ f$
- If f and g are injective, so is $g \circ f$
- If f and g are surjective, so is $g \circ f$
- If f and g are bijective, so is $g \circ f$

1.1.4 Equivalence Relations

Definition 1.1.3: Equivalence Relation

An equivalence relation is a relation that has the following special properties:

- Reflexivity: aRa for all $a \in A$
- Symmetry: $aRb \implies bRa$
- Transitivity: $aRb \wedge bRc \implies aRc$

Definition 1.1.4: Partition

Given a set S, a partition of S is a collection of subsets of S such that their union is S.

Note:

Equivalence relations go hand in hand with partitions.

Note:

If $\sim$ is an equivalence relation $a \sim b$, then $\sim$ partiations a set X into chunks. $X/\sim$ is the set of chunks.
Addition is *well-defined* as an operation on $\mathbb{Z}/x\mathbb{Z}$ for $x \in \mathbb{Z}$.

1.1.5 Complex Numbers and Matrices

Definition 1.1.5: Complex Number

A complex number is a number of the form $a + bi$, where a and b are real numbers and i is the imaginary unit. $i^2 = -1$.

Note:

Complex numbers generally take the from $z = a + bi$.
$\bar{z} = a - bi$ is the complex conjugate of z.

Definition 1.1.6: Matrix

A matrix is a rectangular array of numbers.

1.1.6 Number Theory

1.2 Random Examples

Claim 1.2.1 Topology

Topology is cool

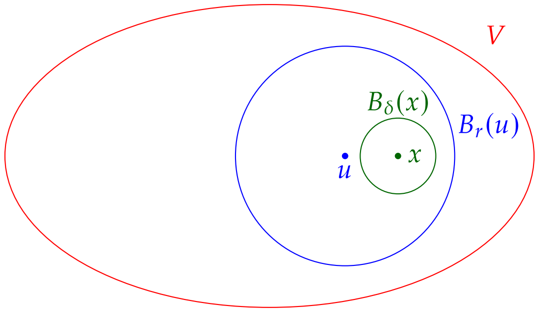
Example 1.2.1 (Open Set and Close Set)

Open Set:
- ϕ
- $\bigcup\limits_{x \in X} B_r(x)$ (Any $r > 0$ will do)
- $B_r(x)$ is open

Closed Set:
- X, ϕ
- $\overline{B_r(x)}$

$x-$axis $\cup$ $y-$axis

Proof: By openness of V, $x \in B_r(u) \subset V$



Given $x \in B_r(u) \subset V$, we want $\delta > 0$ such that $x \in B_\delta(x) \subset B_r(u) \subset V$. Let $d = d(u, x)$. Choose δ such that $d + \delta < r$ (e.g. $\delta < \frac{r-d}{2}$)

If $y \in B_\delta(x)$ we will be done by showing that $d(u, y) < r$ but

$$d(u, y) \leqslant d(u, x) + d(x, y) < d + \delta < r$$

😄

Corollary 1.2.1

By the result of the proof, we can then show...

Lenma 1.2.1

Suppose $\vec{v}_1, \ldots, \vec{v}_n \in \mathbb{R}^n$ is subspace of $\mathbb{R}^n$.

Proposition 1.2.1

$1 + 1 = 2$.

1.3 Random

Definition 1.3.1: Normed Linear Space and Norm $\|\cdot\|$

Let V be a vector space over $\mathbb{R}$ (or $\mathbb{C}$). A norm on V is function $\|\cdot\|\ V \longrightarrow \mathbb{R}_{\geqslant 0}$ satisfying

1. $\|x\| = 0 \iff x = 0\ \forall\ x \in V$
2. $\|\lambda x\| = |\lambda|\|x\|\ \forall\ \lambda \in \mathbb{R}(\text{or } \mathbb{C}),\ x \in V$
3. $\|x+y\| \leqslant \|x\| + \|y\|\ \forall\ x, y \in V$ (Triangle Inequality/Subadditivity)

And V is called a normed linear space.
• Same definition works with V a vector space over $\mathbb{C}$ (again $\|\cdot\| \longrightarrow \mathbb{R}_{\geqslant 0}$) where 2 becomes $\|\lambda x\| = |\lambda|\|x\|$ $\forall\ \lambda \in \mathbb{C},\ x \in V$, where for $\lambda = a + ib$, $|\lambda| = \sqrt{a^2+b^2}$

Special Case $p = 1$: $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_m|$ is clearly a norm by usual triangle inequality.
Special Case $p \to \infty$ ($\mathbb{R}^m$ with $\|\cdot\|_\infty$): $\|x\|_\infty = \max\{|x_1|, |x_2|, \cdots, |x_m|\}$
For $m = 1$ these $p-$norms are nothing but $|x|$. Now exercise
Solution: **For Property 3 for norm-2**

When field is $\mathbb{R}$:

We have to show

$$\sum_i (x_i+y_i)^2 \leqslant \left(\sqrt{\sum_i x_i^2} + \sqrt{\sum_i y_i^2}\right)^2$$

$$\implies \sum_i (x_i^2 + 2x_iy_i + y_i^2) \leqslant \sum_i x_i^2 + 2\sqrt{\left[\sum_i x_i^2\right]\left[\sum_i y_i^2\right]} + \sum_i y_i^2$$

$$\implies \left[\sum_i x_iy_i\right]^2 \leqslant \left[\sum_i x_i^2\right]\left[\sum_i y_i^2\right]$$

So in other words prove $\langle x, y\rangle^2 \leqslant \langle x, x\rangle\langle y, y\rangle$ where

$$\langle x, y\rangle = \sum_i x_iy_i$$

Note:

- $\|x\|^2 = \langle x, x\rangle$
- $\langle x, y\rangle = \langle y, x\rangle$
- $\langle\cdot,\cdot\rangle$ is $\mathbb{R}-$linear in each slot i.e.

$$\langle rx + x', y\rangle = r\langle x, y\rangle + \langle x', y\rangle \text{ and similarly for second slot}$$

Here in $\langle x, y\rangle$ x is in first slot and y is in second slot.

Now the statement is just the Cauchy-Schwartz Inequality. For proof

$$\langle x, y\rangle^2 \leqslant \langle x, x\rangle\langle y, y\rangle$$

expand everything of $\langle x - \lambda y, x - \lambda y\rangle$ which is going to give a quadratic equation in variable λ

$$\begin{aligned}\langle x - \lambda y, x - \lambda y\rangle &= \langle x, x - \lambda y\rangle - \lambda\langle y, x - \lambda y\rangle \\ &= \langle x, x\rangle - \lambda\langle x, y\rangle - \lambda\langle y, x\rangle + \lambda^2\langle y, y\rangle \\ &= \langle x, x\rangle - 2\lambda\langle x, y\rangle + \lambda^2\langle y, y\rangle\end{aligned}$$

Now unless $x = \lambda y$ we have $\langle x - \lambda y, x - \lambda y \rangle > 0$ Hence the quadratic equation has no root therefore the discriminant is greater than zero.

When field is $\mathbb{C}$:

Modify the definition by

$$\langle x, y \rangle = \sum_i \overline{x_i} y_i$$

Then we still have $\langle x, x \rangle \geqslant 0$

1.4 Algorithms

Algorithm 1: what

```
Input: This is some input
Output: This is some output
/* This is a comment */

some code here;
x ← 0;
y ← 0;
if x > 5 then
|   x is greater than 5 ;                         // This is also a comment
else
|   x is less than or equal to 5;
end
foreach y in 0..5 do
|   y ← y + 1;
end
for y in 0..5 do
|   y ← y − 1;
end
while x > 5 do
|   x ← x − 1;
end
return Return something here;
```