CONSTRUCTION OF THE CANTOR ALGEBRA

JACKSON WALTERS

1. Introduction

There are a number of instances in mathematics where we construct an object and ask that it satisfy a few natural properties, and are surprised to learn that there is only one such object up to isomorphism. Consider the real numbers, which are the unique complete ordered field. The complex numbers are the unique algebraically closed field containing the real numbers as a subfield. For each prime power $q = p^a$, there is only one finite field $\mathbb{F}_q$ up to isomorphism. The integers are the initial object in the category of rings, unique up to isomorphism. The rational numbers are the smallest field contained within the real numbers. The algebraic closure $\overline{\mathbb{F}}_p$ is the unique algebraically closed field of positive characteristic. In this way we obtain the familiar hierarchy:

$$\overline{\mathbb{F}}_p \hookleftarrow \mathbb{Z}/p^a\mathbb{Z} \hookleftarrow \mathbb{Z}/p\mathbb{Z} \xleftarrow{\text{mod } p} \mathbb{Z} \hookrightarrow \mathbb{Q} \hookrightarrow \mathbb{R} \hookrightarrow \mathbb{C}$$

We use these familiar numbers systems for almost all the math we encounter. In geometry, manifolds are modeled locally by real numbers. In algebra, we solve characteristic polynomials over the complex numbers to obtain eigenvalues. In cryptography, we consider polynomials over finite fields and perform number theoretic transforms. We have ways to represent these numbers or approximations of them in almost every programming language.

In order to perform computations with these numbers, we require some way to represent them on the computer, so some conversion to sequences of bits. This allows us to perform classical computations on the input bits. It turns out the circuits we write consist essentially of three basic gates: AND, OR, NOT. These gates process one or two bits at a time, and are capable of representing essentially any function you can imagine. These circuits ultimately reduce to expressions in Boolean variables x_i which can take values in a two element set, say $\{0, 1\}$. Often we think of these as representing the mutually exclusive values $\{True, False\}$, which have become synonymous with "Boolean".

We usually write conjunction as $x_i \wedge x_j$, disjunction as $x_i \vee x_j$, and negation as $\neg x_i$, and sometimes refer to $\wedge$ and $\vee$ as meet and join, respectively. With these operations, the collection of n elements $\{x_i\}_{i=0}^{n-1}$ generates an algebraic structure $\mathbb{B}_n$ which is commutative, associative, and idempotent. Idempotency separates this structure from usual numbers, in

Date: February 19, 2026.

that $x \wedge x = x$ and $x \vee x = x$ for all $x \in \mathbb{B}_n$. This structure has 2^n elements. It turns out that every finite Boolean algebra has size which is a power of two, and that these algebras are unique up to Boolean algebra isomorphism. The basic Boolean algebra $\mathbb{B}_1 = \{0, 1\}$ has exactly two elements.

Recall that finite dimensional vector spaces over a field $\mathbb{K}$ are canonically isomorphic to $\mathbb{K}^n$ for some n. This isomorphism is afforded by choosing a basis. Similarly, in the world of Boolean algebras we note that every element is some combination of the $\{x_i\}$ using logical operations, e.g. $\neg x_0 \vee x_1 \wedge x_3 \vee x_2 \wedge x_4$. The generators of the algebra are referred to as *atoms*. Indeed, for any finite Boolean algebra we can consider its set of atoms B_n, which are essentially the elements that cannot be written as joins of other elements. On the other hand, if we consider $\mathcal{P}(B_n)$, the power set of the atoms, or set of all subsets of B_n, we obtain a Boolean algebra using the operations union, intersection, and complement. It turns out that $\mathcal{P}(B_n) \cong \mathbb{B}_n$, and this correspondence is duality between finite Boolean algebras and finite sets.

There is an intuitive way to think of this picture as well. Consider you have the set $\{apple, orange, pear\}$. We could consider the eight element collection of subsets (including the empty set $\emptyset$). We find our set at the bottom of this lattice above $\emptyset$. Rather than using actual apples, oranges, and pears, we might label buckets with stickers of an apple, orange, and a pear. If there's a ball in the bucket, we say that an apple is present, similar for orange and pear. If we have two balls, we can consider the subset, say, $\{pear, apple\}$. Even more abstractly, we might get rid of the stickers altogether, and just label the buckets with integers $\{0, 1, 2\}$. Now we know if there's a ball in bucket 0, that represents an apple, and so on. Thus, if I have three balls at my disposal, I can represent any subset in the collection. I can also perform the operations if I have a few extra balls.

Even more abstractly, we might line up the buckets, and remove the $\{0, 1, 2\}$ labels. This is called positional notation. Now we can simply represent apple as the *bitstring* 100, and orange as 010, and pear as 001. This is about as abstract as it gets, because the only information we are retaining is our ability to distinguish three objects. We can distinguish zero and one, but we cannot distinguish one from one. This is also known as one-hot encoding.

Another way to think about this is to imagine that an apple has very many bits, perhaps on the order of 10^{30} or more. We don't need all those to distinguish it from a pear. The stickers are a first abstraction, removing a ton of information to get a small 2D picture or emoji. Using color would represent a further abstraction, so the set $\{red, orange, green\}$. Going down to bitstrings removes as much information as possible.

Remark. One could argue that actually representing a bit is no trivial matter, as you might need a huge number of electrons stored in a transistor or physical means, but that is a philosophical issue.

Once we are comfortable with bitstrings, we might think of $(x_i)_{i=0}^{n-1}$ as a vector or string of n variables representing possible bits. We can now apply our operations component-wise, so bitwise AND, OR, NOT. A natural question to ask is: what happens when we consider not only finite bitstrings, but infinite bitstrings? This is a natural idea, and leads to the first example of an infinite Boolean algebra.

We have to be slightly more careful when we define this algebra, since now the topology is no longer discrete. In the finite case, we could simply consider the collection of all subsets of the underlying set. In this case, the underlying set is formed using the infinite strings, and the topology is no longer discrete but inherited from the subspace topology of the real line. This space is formed by removing middle thirds repeatedly from a unit interval in the real line. The space turns out to be uncountable, and is called the *Cantor space*. The infinite strings specify how to move down an infinite binary tree to get to a specific point in the space. The Boolean algebra is not this space directly, but rather the algebra of clopen (closed and open) subsets of the Cantor space. This collection of sets is countable, and turns out to have a very intuitive algebraic structure.

However, something slightly mysterious happens as well: the algebra has no atoms; it is *atomless*. It turns out that the *Cantor algebra* is the unique countable atomless Boolean algebra. This paper will attempt to demystify its structure.

2. Construction

The *Cantor algebra* is essentially the Stone dual of the *Cantor set*, the subspace of the unit interval $[0, 1]$ in the real line formed by removing middle thirds repeatedly. That is, we remove the middle third of the interval to obtain the interval $I_1 = [0, 1/3] \cup [2/3, 1]$. Then we remove two segments from each of the remaining thirds to obtain $I_2 = [0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1]$. Let $I_0 = [0, 1]$ and I_n denote the set obtained after n iterations of this process. Then the Cantor set is the limit as $n \to \infty$ which we denote $\mathcal{S} := I_\infty$. Note that the Cantor set contains both 0 and 1, so is nonempty. $\mathcal{S}$ is uncountable by a diagonalization argument.

The set of infinite bitstrings are maps $b : \mathbb{N} \to \mathbb{B}_1$ with bits b_i. For example, one could have the bitstring $b_i = 0$ if i is even and $b_i = 1$ if i is odd, e.g. $010101\ldots$. We can identify the Cantor set with such bitstrings by thinking about a binary tree above the unit interval. Place a node at height 1 above the midpoint, so at $(1/2, 1)$. Note this is in the middle of the first removed interval in the above process. At the second step in the process, put two nodes at height $1/2$ each above the midpoints of the two removed intervals. Connect these two nodes to the first node with an edge. Repeat this process ad infinitum to obtain an infinite binary tree. For each infinite binary string, walk the binary tree downwards going left if $b_i = 0$ and right if $b_i = 1$. The x-coordinate of the limit is a point in the Cantor set. Since this set is a subset of the unit interval, we equip the set with the subspace topology

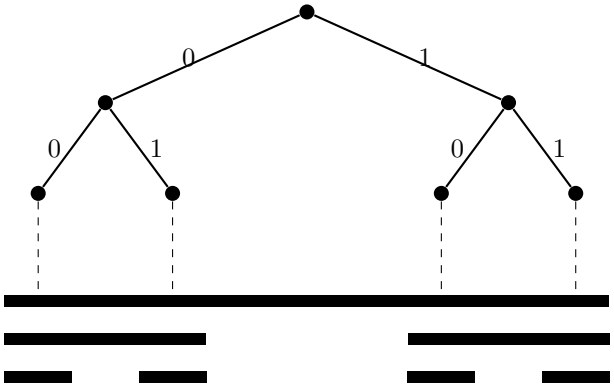

FIGURE 1. The Cantor set is formed by removing middle thirds from a line segment repeatedly.

to obtain the *Cantor space*.

We may now form the *Cantor algebra* as $\mathcal{A} = clopen(\mathcal{S})$, as the collection of clopen (closed and open) subsets of the Cantor set. The clopen sets are formed as finite unions of cylinder sets. Let a *prefix bitstring* s be a map from $\{0, 1, \ldots, n\} \to \mathbb{B}_1$, so a finite string of zeros and ones, e.g. 010101. Note that each binary string can be identified uniquely with an integer using its binary representation, so the set of prefix strings, and therefore cylinder sets, is countable. Since finite unions from a countable set is again countable, $\mathcal{A}$ is countable.

Definition 2.1 (Cylinder Set)**.** Given a finite prefix bitstring s of length n, a cylinder set C_s is the set of infinite bitstrings b such that $b_i = s_i$ for $0 \leq i \leq n-1$.

That is, we fix the prefix s and consider all the infinite bitstrings with any suffix, as long as the prefix agrees. These cylinder sets form a basis for the algebra, in that every clopen set can be written as a finite union of cylinder sets.

Theorem 2.1 (Cylinder sets form basis)**.** *Every clopen set $C \in \mathcal{S}$ can be written as a finite union of cylinder sets $C = \cup_{j=0}^{n-1} C_j$.*

Proof. First note that every cylinder set C_s is open because it is a basic open set in the product topology on $\{0,1\}^{\mathbb{N}}$, which is the topology inherited by the Cantor space. Note that $\mathcal{S} = \bigcup_{s \in S} C_s$ where $S = \{s : len(s) = n\}$. That is, cylinder sets of length n partition the space into a disjoint union of 2^n cylinder sets. C_s is closed because its complement is a finite union of cylinder sets at the same level. Since the cylinder sets form a basis for the product topology, any open set U can be written as a (possibly infinite) union of cylinder sets.

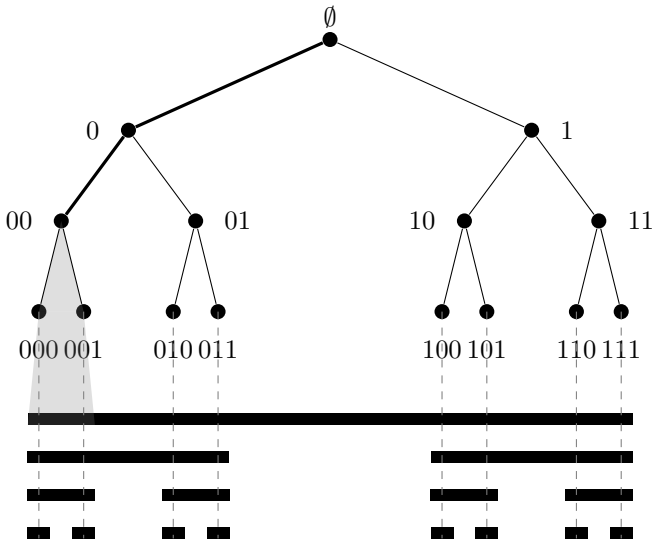

FIGURE 2. The cylinder set C_{00} consists of every infinite string with prefix 00 shown as the shaded region on the left.

Recall that the Cantor set $\mathcal{S}$ is compact as it is a closed and bounded subset of $\mathbb{R}$. Now suppose $C \subset \mathcal{S}$ is clopen. Being open, C can be written as a union of cylinder sets. Being closed in a compact space, C is itself compact. Since C is compact and the cylinder sets form an open cover of C, there exists a finite subcover. Therefore C is a finite union of cylinder sets. □

Definition 2.2 (Compatibility)**.** Two cylinder sets C_s and C_t of length m, n respectively are *compatible* if they agree entirely on the smaller prefix. Suppose WLOG $m < n$, then they are compatible if $s_i = t_i$ for $0 \leq i \leq m - 1$.

Cylinder sets have the nice property that if they are compatible, then their intersection is another cylinder set, and otherwise their intersection is empty.

Lemma 2.2 (Incompatible intersection)**.** *Suppose C_s and C_t are two cylinder sets. If they are not compatible, then their intersection is empty.*

Proof. Since they are not compatible, there must be some index i at which s and t disagree, so suppose $s_i \neq t_i$. If $u \in C_s \cap C_t$, then $u_i = s_i$ and $u_i = t_i$, so $s_i = t_i$, which is a contradiction. Therefore $C_s \cap C_t = \emptyset$. □

If the cylinders are compatible, then their intersection is another cylinder set.

Lemma 2.3 (Compatible intersection)**.** *Suppose C_s and C_t are two cylinder sets. If they are compatible, then their intersection is another cylinder set C_u such that u is the longer of the two strings s, t.*

Proof. Suppose $len(s) = m$ and $len(t) = n$. Since the cylinder sets are compatible, $s_i = t_i$ for $0 \leq i \leq min(m, n) - 1$. Suppose WLOG that $len(s) \leq len(t)$. If $u \in C_s \cap C_t$, then u must agree with s and t. Since t is longer, agreement on t implies agreement on s. Therefore $C_s \cap C_t = C_t$, another cylinder set. □

Given a cylinder set, it is possible to break it apart into two smaller cylinders.

Lemma 2.4 (Cylinder splitting)**.** *Suppose C_s is a cylinder set. Then there exists cylinder sets C_t and C_u such that $C_s = C_u \cup C_t$ where $C_u \subset C_s$ and $C_t \subset C_s$ and $C_u \cap C_t = \emptyset$.*

Proof. Suppose $len(s) = n$. Then we may form the two bitstrings $u := s \cup \{0\}$ and $t := s \cup \{1\}$. Both bitstrings have length $n + 1$, and C_u and C_t are incompatible, so their intersection is empty. Suppose $x \in C_s$. Then x_n is either 0 or 1. If $x_n = 0$, then $x \in C_u$. If $x_n = 1$, then $x \in C_t$. Therefore $x \in C_u \cup C_t$. On the other hand, suppose $x \in C_u \cup C_t$. If $x \in C_u$, then $x \in C_s$ since C_u and C_s are compatible. Similarly for C_t. □

Now let's show that the algebra $\mathcal{A}$ satisfies the properties of being closed under union, intersection, complement.

Lemma 2.5 (Closure under union)**.** *Let $C_0, C_1 \in \mathcal{A}$. Then $C_0 \cup C_1 \in \mathcal{A}$.*

Proof. Since $C_0, C_1 \in \mathcal{A}$, $C_0 = \cup_{i=0}^{m-1} C_{s_i}$ and $C_1 = \cup_{j=0}^{n-1} C_{t_i}$. Therefore $C_0 \cup C_1 = \cup_{i=0}^{m-1} C_{s_i} \cup \cup_{j=0}^{n-1} C_{t_i}$, a finite union of cylinder sets. □

Lemma 2.6 (Closure under intersection)**.** *Let $C_0, C_1 \in \mathcal{A}$. Then $C_0 \cap C_1 \in \mathcal{A}$.*

Proof. Since $C_0, C_1 \in \mathcal{A}$, $C_0 = \cup_{i=0}^{m-1} C_{s_i}$ and $C_1 = \cup_{j=0}^{n-1} C_{t_i}$. Therefore

$$C_0 \cap C_1 = (\cup_{i=0}^{m-1} C_{s_i}) \cap (\cup_{j=0}^{n-1} C_{t_j}) = \cup_{i=0}^{m-1} \cup_{j=0}^{n-1} C_{s_i} \cap C_{t_j}$$

by the distributive property. Recall that the intersection of cylinder sets is either empty or another cylinder set. Thus, if any intersection $C_{s_i} \cap C_{t_j}$ is empty, then we omit it from the pairwise union. Otherwise, we obtain another cylinder set, so we obtain a finite union of cylinder sets. □

Lemma 2.7 (Closure under complements)**.** *Suppose $C \in \mathcal{A}$. Then $C^c \in \mathcal{A}$.*

Proof. Write $C = \bigcup_{i=0}^{m-1} C_{s_i}$, a finite union of cylinder sets. Let N be the maximum of the lengths of the s_i. Refining to level N, each C_{s_i} can be written as a finite union of cylinder sets of length N. Hence there exists a finite set $S \subseteq \{0,1\}^N$ such that

$$C = \bigcup_{t \in S} C_t.$$

Since the sets $\{C_t : t \in \{0,1\}^N\}$ are pairwise disjoint and partition the space, we have

$$C^c = \left(\bigcup_{t \in S} C_t \right)^c = \bigcup_{t \notin S} C_t.$$

Thus C^c is a finite union of cylinder sets, so $C^c \in \mathcal{A}$. □