The Mathware Project:
The Space of Binary Sequences
Part 1

Julia Gibson

May 2023

1 What To Expect From This Project

You can expect each installment of this project to contain some actual math and some actual code. At minimum, there will be code and math that addresses a coding or math question posed in the immediately preceding installment. My personal goal for these questions is that they will be 1) instructive or enlightening, and 2) novel relative to what I can find covered on the internet. You can also expect the language surrounding that actual math and actual code to be rigorous—but not stiff! You can expect me to be considerate of the fact that math and code are difficult objects to understand and create, even for professionals, and even if they make out as though they never have difficulty with them. They are lying!!

2 Question: Where To Begin?

What is the shape of the intersection between mathematics and computing? An intriguing (probably ill-posed!) question that at least *seems* to require being able to identify the contents of that set. Wow, that sounds hard! I am not going to attempt to answer this question, even though I find it inspiring. Instead, I'm going to do something much more tractable: pick a topic I know to be in that intersection and start working with it.

3 Answer: Just Begin Wherever You *Will* Begin!

So, what is the shape of the intersection between mathematics and computing? No idea! I do know that I want to explore that space. I *suspect* that exploring this space will result in some insight into its shape in much the same way that exploring the Earth as a mariner gives you useful data with which to make

inferences about the shape of the Earth. Okay, so where is a good entry point into this space? Well, if I spend too long thinking about that question, I risk never actually getting my hands dirty doing what I set out to do, so best just to sidestep that entirely and just start *somewhere* in that space. I do at least know that part of what gives that intersection whatever shape it has is the set of all binary sequences. So! Let's get our hands dirty.

4 Getting Clarity: The "What"

We need to get a handle on *exactly* what we mean when we refer to "the set of all binary sequences". As we'll see, there is a lot of richness to be gained from digging into the details of this object as viewed from different perspectives.

4.1 Binary Sequences in Computing

The set of all binary sequences is in some sense completely natural and familiar to us in the context of computing: If we know a little bit about how modern computers work, we know that the objects over which the hardware inside a conventional desktop machine is operating are representable as sequences of 1s and 0s — i.e. of binary digits.

A natural question is whether, by using such simple objects as binary sequences, we sacrifice the richness of representation that, say, decimal sequences affords us. The follow-up question is whether there is a map to take us from any given decimal sequence to a corresponding unique binary sequence, and vice versa. There are some subtleties, but basically the answer is 'yes'; we can travel back and forth at will between these representation systems, and so which one we use in practice is up to practical considerations.

Aside 1. *This is one of the awe-inspiring characteristics of technology in our epoch: that we can use mental abstractions with such simple components and build very sophisticated tools for encoding information that can be used to generate new knowledge or new sensory experiences—say, by allowing us to display complex imagery on LCD displays controlled by a binary computer.*

As simple as binary sequences and their building blocks seem, let's define these abstractions. First, what exactly do we mean by "sequence"? Let's keep in mind the informal notion that a sequence is an ordered list. This is the idea that we want to make precise so that we can engage with sequences more easily and reliably. Now, a motivating example of this idea:

Example 1. *Consider the list of integers 1, 11, 21, 1211, 111221, 312211, 13112221,... If we thought of this as just a* set *of numbers, which, by the definition of a set, has no ordering, then we would consider the list 11, 1, 1211, 21, 111221,... or any other permutation of the original list to be equal to the first list above.*

We're allowed to forget about order of list entries whenever we want and just deal with sets, but if we do that we're implicitly talking about different sequences. In many situations, order really does matter and we want to keep track of the indexing data that tell us the ordering of list entries.

In the above example, order is important because the most natural rule for generating sequence entries is *recursive*, meaning it defines the next term in the sequence with respect to at least one term preceding it. 'Preceding' requires that an ordering is part of the list structure.

Aside 2. *The sequence above, if you're not already familiar, is called the Look-and-Say sequence. Can you figure out why?*

With that motivating example in hand, let's introduce the formal definition of a *binary* sequence (which the above example is not).

Definition 1. *A* binary sequence *is a function whose domain is the set of natural numbers, written* $\mathbb{N}$*, whose codomain/range is the set* $\{0, 1\}$.

Notation 1. *We would like both convenient and unambiguous shorthand for both a given sequence and the set* $\{0, 1\}$.

1. The formal name for the set $\{0, 1\}$ *in mathematics is* $\mathbb{Z}/2\mathbb{Z}$*, which people often read as, "zee-mod-two-zee". You can consider this to be the set where all even integers are identified with 0 and all odd integers are identified with 1.*

2. Even though a binary sequence is just a particular type of function and we could legally notate a given sequence, call it f*, by the usual convention as* $f : \mathbb{N} \mapsto \mathbb{Z}/2\mathbb{Z}$*, in practice, it's usually notated* $(f)_n$ *or* $\{a_n\}$*, where* a_n *is an individual term in the sequence, i.e. function value. I will stick with* $\{a_n\}$ *throughout.*

Example 2. *An important class of binary sequence in applied computing is that with format ...*11111111*...*1*. The exact quantity of '1's depends on the particular computer hardware in use. In computing standard IEEE-754 (which has very widespread use), when this sequence represents the order of magnitude of a binary-encoded number (i.e. is the representation's "exponent") and the digits of the number without order of magnitude digits (i.e. the representation's "mantissa") is 0, this binary sequence signifies signed infinities*[1]*; even if we can't actually store an infinitely large value, we* can *store a representation of the* concept *of such a thing, which is neat!*

In the above example, notice the key property that our sequence *ended*! This is, in my mind, *the* critical distinguishing feature of the space of binary sequences for practical computing and the theory thereof. The reason is that storing a binary sequence is a physical operation, and the hardware that performs this operation is finite in capacity. In particular, in practice, much depends on the length of the longest atomic binary sequence that a given machine can store.

This fundamental length can be a very useful constraint in considering problems in computing, such as error detection and correction in memory. Suppose a memory location on, say, your laptop, can store a single binary sequence 64 bits in length. How corrupted could this sequence possibly get? What's the optimal process for correcting a given corrupt string? Well, our answers depend on how we quantify things in the space of binary sequences 64 bits in length—for example, how we measure how *far away* two arbitrary sequences in this space are. There isn't a unique way to define this notion of distance in this space, at least without introducing more constraints on exactly what type of measurement we're looking for. For instance, a famous distance metric on binary sequences is the *Hamming* distance, which is defined as the number of corresponding bit positions in which two binary sequences of the same length differ. I'll say much more about this and other distance metrics on the space of binary sequences in the next *Mathware* installment.

4.2 Binary Sequences in Mathematics

Suppose we wanted to make some generalizations about binary sequences that would be useful to know and apply when working with modern computing systems. We would be handcuffing ourselves to impose the limitation of referring only to *particular* binary sequences. The tool we need is language to refer to various collections, i.e. sets, of binary sequences and their properties (In wielding this tool, we will not necessarily be restricted to thinking only about finite binary sequences, though oftentimes it might be more useful to do so.).

Let's begin by considering the set of *all* binary sequences, both finite and infinite in length, which I've seen usually notated in the literature as $\{0,1\}^N$. This means that we allow into our space sequences as short as 0, which has length 1, and sequences as long as the alternating sequence $0,1,0,1,0,1,0,...$, which has a compact *description* but never terminates and so has (countably) infinite length. What properties does this space have? And, before going further, do we want to refine exactly what type of space we want to consider it as? For example, we could consider it as a vector space, metric space, as simply an algebraic object such as a ring, etc. I know just enough to think it a useful course of action to consider the specific cases of finite binary sequences (of arbitrary length) and infinite binary sequences separately. In that vein, our first question: How is this space similar and distinct from the space of integers, $\mathbb{Z}$? To make meaningful progress on this question, let's just think about both of these sets as algebraic objects. We could construct a ring isomorphism between the set of *finite* binary sequences and $\mathbb{Z}$, starting as follows:

Claim 1. $(\{0,1\}^N, +, *) \cong (\mathbb{Z}, +, *)$ *as rings, where if* $a, b \in \{0,1\}^N$*, then define*

$$a + b = \{a_i + b_i\}_n$$

with the usual place-value carry operation and

$$a * b = \underbrace{(\{a_i\}_n) + (\{a_i\}_n) + \ldots + (\{a_i\}_n)}_{b \ \ times}$$

Proof. We could get a bit pedantic here and go through all the steps of defining a map $f : (\{0,1\}^N, +, *) \cong (\mathbb{Z}, +, *)$ and showing that f is a bijective ring homomorphism. The essence of the argument that I hope to convince you of with mere handwaving is that, once we've erased the commas in the sequence-representations for a and b to end up with bijectively-corresponding binary strings a' and b', a' and b' just *are* integers written in non-standard, i.e. non-decimal, notation. How do we know this, show this? In brief, my answer is that we show that binary strings and decimal strings are equivalent as *models* of the set of axioms that are accepted within the mathematical community as defining $(\mathbb{Z}, +, *)$. A less rigourous approach is to simply appeal to the binary-to-decimal map

$$a_2 = (\sum a_i)_{10}$$

□

But what about the subset of all binary sequences that have countably infinite length? In particular, does it share the above correspondence with a subset of the integers? If we start with an arbitrary non-terminating binary string, e.g. 11001100110011001100... can we find a terminating decimal representation of that string given by any bijective ring homomorphism?

I'm sweeping a lot of details aside here, but basically the answer is *NO*. The subset of all binary sequences that have countably infinite length is one formulation—and, in set theory, the canonical formulation—of a *Cantor space*[2]. In the next installment I will go into more details about the geometric and topological properties of this type of space. For now, suffice it to say that this subset of binary sequences has a fundamentally different character than the subset of binary sequences with arbitrary finite length.

4.3 Brief Bonus: Binary Sequences in Neuroscience

Binary sequences in neuroscience can be used as a formal approximation of neural spike trains. In theoretical neuroscience, we abstracts away the biological particulars of cells to varying degrees based upon the question of interest. A common abstraction ignores the nuances of neural activation and just puts a binary value on the state of neuron in the following way: assign the neuron a state value of '1' when it is firing and a state value of '0' when it's not firing. Now, if we consider the state of a neuron evolving over time, we can model that time series as a sequence of firing/not firing states, e.g. $0, 0, 0, 1, 0, 0, 0, 1, 1, ...$ We can even model a whole population of neurons over the same time window as a "stack" of sequences of the same length—i.e. as a matrix! From here, there

is an extensive toolset from linear algebra and associated areas of mathematics with which we can work to build testable theories of neural behavior.

$$\begin{bmatrix} 0,0,0,1,0,0,0,1,1 \\ 1,1,1,1,0,0,0,0,0 \\ 0,1,0,0,1,1,1,0,0 \end{bmatrix}$$

In this example, we could consider the rows of the above matrix to either represent the firing time series (a.k.a. "spike trains") of three distinct neurons or the time series of the same neuron over three distinct time trials.

5 Finding The Frontier of Knowledge

One strategy for finding what we think of as the frontier of research in a given field of study, and, in fact, the one I've seen the most frequently in academic settings as a student, is to rely on people paid to create new knowledge to tell you what's both of interest to the academic community and hasn't been found out yet. A different approach is to ask a question yourself and search for any evidence of work on it across whatever published material you can get your hands on. A third is to capitalize on others who themselves have opted for option two.

For this first installment, I am going to cheat and simply present a problem that I think paves the way nicely for the next installment's topic and that does not obviously have lots of solutions posted on the internet after a quick round of Google searching.

6 A Mathware Kick For You:

Given a binary string S of finite length n and a Hamming-distance value m, write a program that can generate all the binary strings within Hamming-distance m from S (That is, compute the "Hamming-ball" around the point S in the space of binary strings of length at most n.).

7 Resources

A truly delightful website for math nerds:
The Online Encyclopedia of Integer Sequences: https://oeis.org/

8 References

1. https://people.orie.cornell.edu/snp32/orie_6125/ieee754/ieee754.html
2. https://web.math.utk.edu/ freire/teaching/m467f19/Cantor$_{S}paces_{T}opology.pdf$