Math 29: Applications of Computability Theory

April 29th, 2022

1 Groups

Recall that a **group** is a set X together with a function $* : X^2 \to X$ such that

- There is an e in X such that $e * x = x * e = x$ for all $x \in X$. That is, there is an identity element.
- For all $x \in X$, there is $y \in X$ such that $x * y = y * x = e$. That is, every element has an inverse.
- For all x, y, and z, $(x * y) * z = x * (y * z)$. That is, $*$ is associative.

The canonical example is the group of integers under addition. Notice that this group is **computable**. What this means is that the group operation $*$ is computable as a function on natural number codes for elements of the group. (That is, the function $*_{\#} : \omega \to \omega$ such that $*_{\#}(\langle \#x, \#y \rangle) = \#(x * y)$ is computable.)

Any finite group is computable, of course. Other examples include the free group, rational numbers under addition, and the rational, invertible $n \times n$ matrices. The real numbers, and real invertible matrices, are not computable in this sense because they are uncountable. However, they are "computable" in the sense we will describe later.

A group **presentation** is a collection of **generators** and **relations**. Every element can then be written as a product of powers of generators. Relations provide rules which allow us to simplify multiplications. For example, if a is the only generator and there are no relations, then the generated group is just the integers. If we add the relation which specifies $a * a = e$, then the group is $\{e, a\}$.

Given a group presentation, the **word problem** for the group is the problem of determining whether or not two products of generator powers are equivalent. In other words, it is the problem of simplifying group multiplication. As we will see next time, even for computable or c.e. groups, the word problem need not be solveable.

2 Analysis

A real number r is said to be **computable** if there is a computable function f such that, for any rational $p > 0$, $f(\#p) = \#q$, where q is a rational number with $|r - q| < p$. That is, p is ϵ, and q is δ. However, we use different letters to make clear that they must be rational.

This mostly aligns with our intuition. All integers and rationals are computable. Numbers like π and e are computable: By Euler,

$$\frac{\pi^2}{6} = \sum_{i=1}^{\infty} \frac{1}{i^2}$$

Thus π can be approximated by first approximating π^2 very closely as $\Sigma_{i=1}^{\infty} \frac{6}{i^2}$, then applying the "long square root" algorithm will allow us to compute the square root of the approximation to π^2, which will approximate π. e is even easier to approximate, as it is the limit of $(1 + \frac{1}{n})^n$.

Notice that, as with functions and sets in the natural numbers, there are countably many machines, and therefore countably many computable reals. **In fact, given any noncomputable set X, the real in the unit interval whose binary expansion is given by X is not computable.** If it were, then we could compute the original set.

The computable reals form a field, but are not complete as a metric space: we know from classical mathematics that completions are unique, and that $\mathbb{R}$ is the completion of the rational numbers. Therefore, the computable reals cannot be complete, as their completion is also $\mathbb{R}$.

3 Graphs and Trees

Given a graph G with finitely- or countably-many nodes coded by natural numbers, we say G is computable if its edge relation is. It is not hard to see that there are graphs which are computable and graphs which are not. For example, given any noncomputable set C, define G_C to be the graph with an edge between 0 and $n + 1$ if and only if $n \in G$, and no other edges.

In computability theory, **trees** are a very important object of study. A tree is a connected, acyclic graph. In other words, there is exactly one path between any two nodes in the tree. We will specifically be concerned with **Baire Space**, denoted by ω^ω, and **Cantor Space**, denoted by 2^ω. Baire space is the set of all infinite strings of natural numbers, and Cantor space is the set of all infinite binary strings.

We can introduce a topology on both sets using the **finite** strings. $\omega^{<\omega}$ and $2^{<\omega}$ are the sets of finite strings of natural numbers and finite binary strings, respectively. Given $f \in \omega^\omega$ or $X \in 2^\omega$, for $\sigma \in \omega^{<\omega}$ and $\tau \in 2^{<\omega}$ we write $\sigma \preceq f$ ($\tau \preceq X$) to mean that $f(n) = \sigma(n)$ ($\tau(n) = X(n)$) for all $n < |\sigma|$ ($n < |\tau|$). We say f (X) extends σ (τ). Then given $\sigma \in \omega^{<\omega}$, $[\sigma] = \{X \in \omega^\omega : \sigma \preceq X\}$, and similarly for 2^ω. In other worse, given a finite string, it generates the basic open set of all functions/sets which agree with it.

$[\sigma]$ will be **clopen** - both open and closed. It is open, as a basic open set, but it will be closed because its complement will be the union of all the basic open sets whose final element disagrees with the final element of σ. This topology will be **compact** and **totally disconnected** - it has no nontrivial connected subsets. In the case of Cantor space, under the identification between sets and real numbers in the unit interval given by binary expansions, this gives a totally disconnected topology on the reals. Cantor space is so-named because this topology corresponds to the topology generated by the Cantor middle thirds set.

A **binary tree** T is a subset of $2^{<\omega}$ which is closed under initial segments: if $\sigma \in T$ and $\tau \preceq \sigma$, $\tau \in T$. Then

$$[T] = \{X \in 2^\omega : \text{For all } n, \text{ the first n bits of X are in } T\}$$

We call elements of $[T]$ paths of T. Notice the following basic facts about binary trees.

Lemma 1. *If T is an infinite binary tree, then it has an infinite path.*

Notice that this is not true for Baire space: the tree containing the empty strings and all strings of length 1 is infinite, but it clearly has no infinite path.

Importantly, this fact is not computably true: there is an infinite, computable tree with no infinite, computable path.

Lemma 2. *Given any two disjoint c.e. sets A and B, the sets which separate A and B form a computable tree.*

We know (or will know) from the Take-Home Midterm that there are disjoint c.e. sets A and B which cannot be computably separated. Therefore, the tree constructed above for A and B is an infinite computable tree with no computable paths.