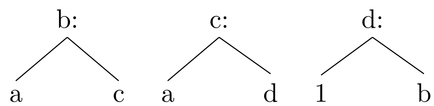
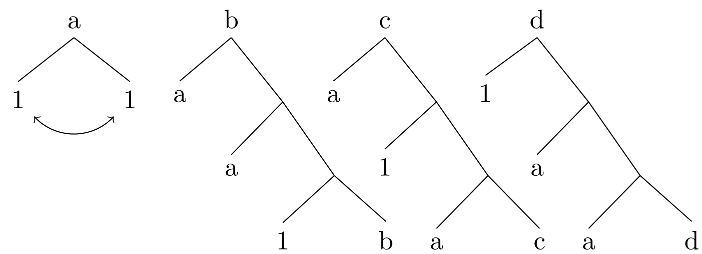
1 Introduction

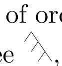
These notes describe a technique for using cyclic linear codes to produce highly self-similar branch groups. These groups should be viewed as generalizations of the first grigorchuk group, which corresponds to the $[3, 2]$ code[1] $\{110, 101, 011, 000\}$ over $\mathbb{F}_2$.

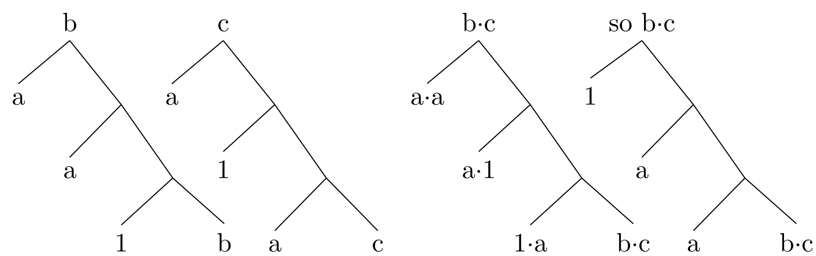
The grigorchuk group is generated by automorphisms of a, b, c, d of the complete binary tree, where a flips the first level, and b, c, d are in the stabilizer the first level, defined by:



It's helpful to spell out this recursion as in



b, c, and d all have order 2, and generate a subgroup of order 4. This is easy to check explicitly: since all of them stabilize the subtree , we can just multiply them pointwise. I'll just show one example



Notice $b \cdot c$ satisfies the same recurrence that d does, so $b \cdot c = d$.

We can identify b, c, and d with the vectors $110, 101$, and 011 respectively, based on the pattern of 1's and a's we see[2]. If we do that, the computation $b \cdot c = d$ amounts to saying $110 + 101 = 011$.

[1] A $[n, k]$ linear code over $\mathbb{F}$ is a k-dimesional subspace of $\mathbb{F}^n$. I will give more details soon

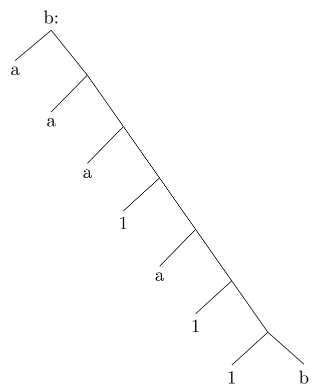
[2] There's an unfortunate switch from additive to multiplicative notation, so $a \mapsto 1$ and $1 \mapsto 0$

2 A new example

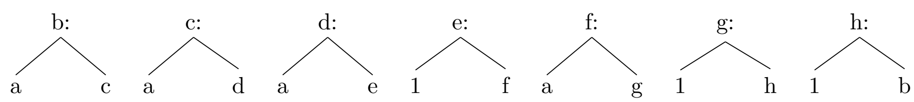
Here is a new example I will call G, based on a cyclic [7,3] code. This is also a group of automorphisms of the binary tree, generated by a and:

b	: 1110100
c	: 1101001
d	: 1010011
e	: 0100111
f	: 1001110
g	: 0011101
h	: 0111010

So, for example:



Similarly to the grigorchuk group, b, c, d, e, f, g, h all have order 2, and generate a subgroup of order 8, isomorphic to $(\mathbb{Z}_2)^3$. As before, this ammounts to just checking the codewords are closed under sums. For reasons I explain next section, this follows from the fact $x^4 + x^2 + x + 1 \mid x^8 - 1$ over $\mathbb{F}_2$. Furthermore, as this code is closed under cyclic permutations, we also have a short recursive presentation for the group:



This group has many properties in common with the classical grigorchuk group, proved in very much the same way

Theorem 2.1. *G is a self replicating 2-group of intermediate growth (self replicating meaning every vertex stabilizer is isomorphic to the original group)*

As this proof has no new content, I'll leave this as a special case of a later result.

Question 1. Is the growth function of G slower than the growth of the first grigorchuk group?

I originally constructed G hoping this would be true; 3/7 generators for G get rewritten as 1 in the next level, as opposed to only 2/3

3 Cyclic codes

An $[n, k]$ linear code C over $\mathbb{F}$ is a k-dimensional subspace of $\mathbb{F}^n$. One refers to elements of C as **codewords**. Traditionally, $\mathbb{F}$ is a finite field ($\mathbb{F}_{2^n}$ in most computer applications), and C is constructed so distinct codewords differ in many coordinates. The minimum number of coordinates where two codewords differ is called the **minimum distance** of the code. As the difference of codewords is another codeword, this is equal to the minimum number of non-0 coordinates in a non-0 codeword, or the **weight** of that codeword.

Example 3.1. The subspace $C = \{110, 101, 011, 000\}$ of $(\mathbb{F}_2)^3$ is a [3,2] linear code with minimum distance 2. A typical application of such a code is as follows: Alice wants to send Bob a 2-bit message over a noisy channel which has a 10% chance to flip each bit. If Alice just sends 2 bits, there is a 19% chance Bob recieves the wrong message.
Instead, suppose Alice sends 3 bits, and Bob knows Alice will always send a word in C. Then there is a 72.9% chance Bob receives the correct message, but only a 2.6% chance Bob recieves the wrong message. 24.4% of the time, the word Bob receives is not in C, and he knows an error occurred; This is often much better than having an incorrect message.

A code is called **cyclic** if every cyclic shift of a codeword is also a codeword; so, if 1110 is a codeword, then so are 1101, 1011, and 0111. Of course, these 4 vectors generate $\mathbb{F}_2^4$, so there are no non-trivial cyclic codes over $\mathbb{F}_2$ containing 1110. Interesting cyclic codes do exist; the code in example 3.1 is cyclic. I will now describe a way to find them.

Suppose C is a cyclic code in $\mathbb{F}^n$. Let's identify $\mathbb{F}^n$ with the ring $R = \mathbb{F}[x]/(x^n - 1)$, using the standard basis $(1, x, \ldots, x^{n-1})$. Suppose $a = (a_0, \ldots a_{n-1}) \in C$. Multiplying by x, we see:

$$\begin{aligned}
x * (a_0, \ldots a_{n-1}) &= x * (a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}) \\
&= a_0 x + a_1 x^2 + \cdots + a_{n-1} x^n \\
&= a_0 x + a_1 x^2 + \cdots + a_{n-1} \cdot 1 \\
&= a_{n-1} + a_0 x + a_1 x^2 + \cdots + a_{n-2} x^{n-1} \\
&= (a_{n-1}, a_0, \ldots, a_{n-2})
\end{aligned}$$

So, $x \cdot a$ represents a cyclic shift of a, so the condition C is cyclic means C is closed under multiplication by x. In other words, C is an ideal in R.

Ideals in R correspond to ideals in $\mathbb{F}[x]$ containing $(x^n - 1)$. Since $\mathbb{F}[x]$ is a PID, these correspond precisely to factors of $x^n - 1$ over $\mathbb{F}$. The monic polynomial that generates a cyclic code (as an ideal) is called it's **generator polynomial**.

Example 3.2. Let's classify dimension 3 cyclic codes over $\mathbb{F}_2$. They must correspond to factors of $x^3 - 1$. Over $\mathbb{F}_2$, this factors as $(x + 1)(x^2 + x + 1)$, so there are 4 possible gnerator polynomials:

1 (1) is everything, so this code is all of $\mathbb{F}_2^3$

$x+1$ This generates the code from example 3.1. You can see this a few ways:

- $x+1|p \Leftrightarrow p(1)=0$. Over $\mathbb{F}_2$, this is the same as having an even number of 1's, which characterizes {110, 101, 011, 000}
- The next lemma will show this is a 3-1=2 dimensional subspace of $\mathbb{F}_2^3$, so it must be generated by $x+1$ and x^2+x.

x^2+x+1 This code is {111, 000}. As $x(x^2+x+1)=x^2+x+1$, so there's only one (linear) generator

Lemma 3.1. *The dimension of a cyclic code in $\mathbb{F}^n$ is $n-k$, where k is the degree of it's generator polynomial.*

Proof. Let C be generated by the polynomial $g(x)$. Then $C=\{f(x)g(x)|f \in \mathbb{F}[x]\}/(x^n-1)$ is generated by $\{x^n g(x)\}_{n\in\mathbb{N}}$. The polynomials $g(x)$, $xg(x)$, $\ldots, x^{n-k-1}g(x)$ are all $\mathbb{F}$-linearly independent in R; you can see this from their leading coefficients. We need to show all higher powers are redundant.

Letting $(x^n-1)/g(x)=x^{n-k}+p(x)$, we have $x^{n-k}g(x)+p(x)g(x)=0$ modulo x^n-1. Since $\deg(p(x))<n-k$, this gives x^{n-k} as an $\mathbb{F}$-linear combination of $g(x)\ldots x^{n-k-1}g(x)$, so $x^{n-k}g(x)$ is redundant. Multiplying both sides by x and replacing $x^{n-k}g(x)$ with lower powers, we see x^{n-k+1} is redundant as well. Inductively, the polynomials $g(x), xg(x), \ldots, x^{n-k-1}g(x)$ suffice to generate C. □

4 Using fields other than $\mathbb{F}_2$

My next goal is to describe a machinery for using codes over arbitrary fields to construct branch groups. I think this will be easiest if we picture a few particular examples, but the general definition should be clear.

The first example is over $\mathbb{F}_3$. We'll show this is a 3-group of intermidiate growth. This group is not actually new; it is isomorphic to a special case of the p-groups described in Grigorchuk (1986), where ω is taken as 01233210 repeating.

Example 4.1. Let C be the [8,2] cyclic code over $\mathbb{F}_3$ with generator polynomial $x^6+2x^5+2x^4+2x^2+x+1$ (which happens to divide x^8-1). This was chosen as the smallest cyclic code over $\mathbb{F}_3$ so that every code word contains a 0. C is 2-dimensional over $\mathbb{F}_3$, so it has 9 words in it. The 8 non-0 words correspond precisely to the 8 cyclic shifts of 11202210.

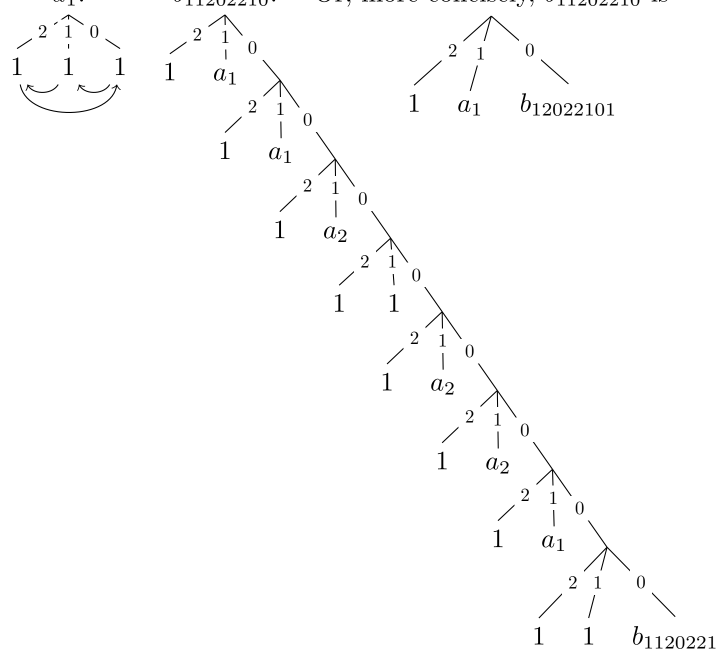
Let $T=\mathbb{F}_3^{<\omega}$ be the 3-regular tree. We build a group G_C of automorphisms of T generated by the following automorphisms:

1. For each $x\in\mathbb{F}_3$, let a_x be the automorphism of T that adds x on the first level; that is $a_x(x_0,x_1,\ldots,x_n)=(x_0+x,x_1,\ldots,x_k)$. This is analogous to a in the grigorchuk group.

2. For each $c = c_0, \dots, c_n \in C$, we define an automorphism b_c analogous to b, c, or d in the grigorchuk group; b_c acts trivially on words not containing 2; otherwise $b_c(x_0, x_1, \dots 0, 1, x_{m+1}, \dots x_k) = (x_0, x_1, \dots 0, 1, x_m + c_{m\%n}, \dots x_k)$, where m is the level of the first 1 and $\%$ is the modulus operator.

a_0 and $b_{0\dots0}$ are both the identity, so we could leave them out if desired. Here are some sample portraits of other generators

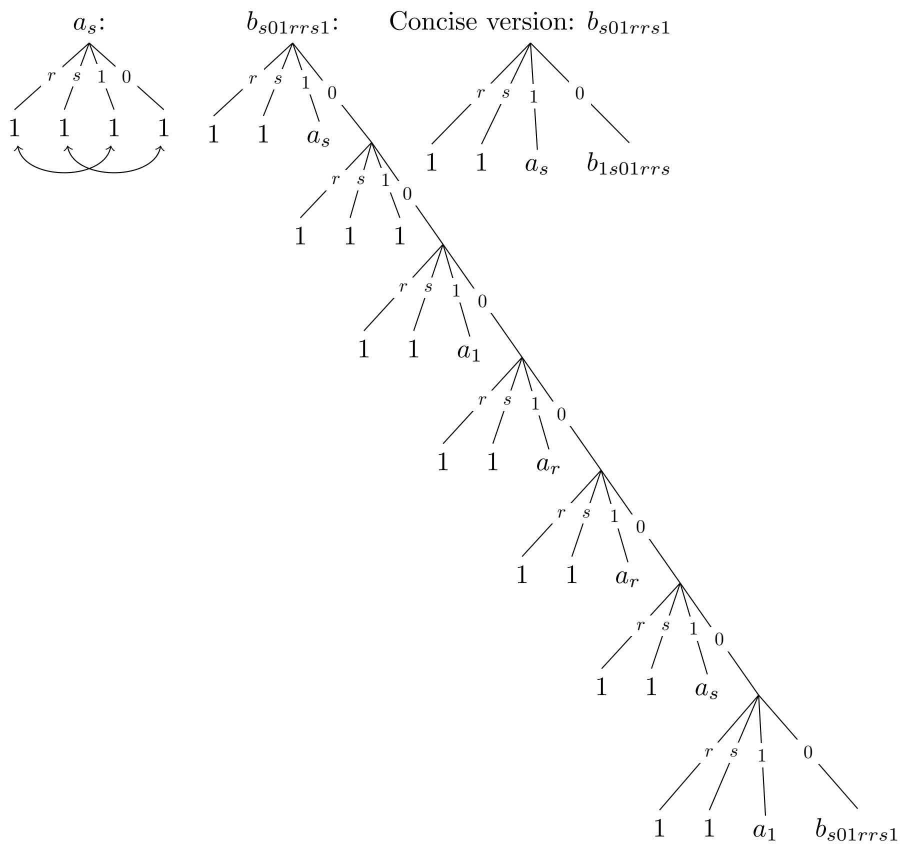
a_1: $\quad b_{11202210}$: $\quad$ Or, more concisely, $b_{11202210}$ is



Applying this construction over non-prime fields is not any harder. Here's an example is over $\mathbb{F}_4$:

Example 4.2. Let $\mathbb{F}_4 = \{r, s, 1, 0\}$, where r and s are the distinct roots of x^2+x+1 (I will always use this order). Let C be the $[7, 3]$ cyclic code generated by 1110100. This is the same generator as used in section 2, but over the larger field, we have words like $r(1001110) + 1(1010011) = (s01rrs1)$. Again, this was chosen as the smallest code over $\mathbb{F}_4$ such that every code word contains a 0.

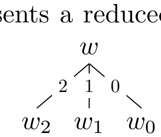
Here are portraits of a_s and $b_{1rrs1s0}$:



It's easy to see $a_x \cdot a_y = a_{x+y}$ and $b_c \cdot b_{c'} = b_{c+c'}$. So, much like the classical grigorchuk group, we can reduce every word in these generators to a word of the form:

$$a_{x_0} b_{c_0} a_{x_1} b_{c_1} \cdots a_{x_n} b_{c_n} a_{x_{n+1}}$$

where each x_i and c_i is non-0, except possibly a_0 and a_{n+1}.

We also have analogous rewriting rules; over $\mathbb{F}_3$, if w represents a reduced word which stabilizes the first level, then w has a portait like ,

or (w_2, w_1, w_0), where w_0, w_1, and w_2 are smaller words. Over $\mathbb{F}_4$, we could similarly get (w_r, w_s, w_1, w_0) . As a base case, remark that this holds for $a_x b_c a_x^{-1}$. For example, $a_1 b_{11202210} a_1^{-1} = (b_{12022101}, 1, a_1)$, or $a_s b_{s01rrs1} a_s^{-1} = (a_s, b_{1s01rrs}, 1, 1)$ in $r, s, 1, 0$ order. We'll work over $\mathbb{F}_3$ for concise notation. The fact C is cyclic guaruntees each component is another element of G_C.

We can decompose any w as products of such conjugates. To see this, let

$x'_0 = x_0$ and $x'_n = x_n + x'_{n-1}$. Then:

$$\begin{aligned} w &= a_{x_0} b_{c_0} a_{x_1} b_{c_1} \cdots a_{x_n} b_{c_n} a_{x_{n+1}} \\ &= a_{x_0} b_{c_0} (a_{x_0}^{-1} a_{x_0}) a_{x_1} b_{c_1} \cdots a_{x_n} b_{c_n} a_{x_{n+1}} \\ &= (a_{x'_0} b_{c_0} a_{x'_0}^{-1}) a_{x'_1} b_{c_1} \cdots a_{x_n} b_{c_n} a_{x_{n+1}} \\ &= (a_{x'_0} b_{c_0} a_{x'_0}^{-1}) a_{x'_1} b_{c_1} (a_{x'_1}^{-1} a_{x'_1}) \cdots a_{x_n} b_{c_n} a_{x_{n+1}} \\ &= (a_{x'_0} b_{c_0} a_{x'_0}^{-1})(a_{x'_1} b_{c_1} a_{x'_1}^{-1}) a_{x'_1} \cdots a_{x_n} b_{c_n} a_{x_{n+1}} \\ &= (a_{x'_0} b_{c_0} a_{x'_0}^{-1})(a_{x'_1} b_{c_1} a_{x'_1}^{-1})(a_{x'_1} \cdots)(a_{x'_n} b_{c_n} a_{x'_n}^{-1}) a_{x_{n+1}} \end{aligned}$$

The action of w on the first level is given by $a_{x_1} \cdots a_{x_{n+1}}$, so $a_{x'_n}^{-1} a_{x_{n+1}}$ is trivial if w stabilizes the first level, and we can ignore the last term. Now, rewrite each conjugated b_{c_n} as a triple, and we let w_2, w_1, and w_0 be the products of the respective components, omitting all 1's and combining adjacent a's or b's to get reduced words. Since each b_{c_n} is rewritten as a b-generator in exactly one component, the total number of b-generators in w_0, w_1, and w_2 is less than w, so the rewritten words are indeed shorter.