Abstract

This paper will use the heuristics of a finite Boolean algebra with operators to examine (using representation theory) a restricted family of setwise stabilizers of 2-modular representations with which polynomial time algorithms might be possible. The key virtue of the approach will be to use single-voice musical passages to help the search.

1 Introduction

Throughout this article, when it's clear that a field is being expected, I will denote GF(k) by k. Also, I will use $\{n\}$ in matrix and Kronecker-delta indices to denote a list of elements, so that $a_{i\{n\}} = (a_{i1}, a_{i2}, \ldots, a_{in})$. (Singletons will be clear by context). In addition, $\{n\}$ will denote $\{1, \cdots, n\}$, and in the cases where convention will call to start at 0, I'll use the ordinal notation $n = \{0, \cdots, n-1\}$ An n-bit-vector v of atoms in a BA 2^n will correspond to the finite set of atoms it represents, which I'll call $S : 2^n \to \mathscr{P}(\{n\})$, and it will be understood that I'm not talking about a singleton. Also $\{n\}$ will be treated as both a set and a list.

Let n be some natural number, and consider a finite Boolean algebra 2^n with normal, completely distributive operators $f_\xi, \xi \in \Xi$. It's easy to see that the automorphism group of the universe of 2^n is isomorphic to S_n, and then that raises the question of which permutations on the atoms induce which changes in the operators. The case of seeing which automorphisms respect the operators can be determined by looking at the automorphism group of the full structure (not just the universe) and won't be treated of here, as it is probably treated of elsewhere. However, if we examine which permutations of the atoms map one operator to another (assuming a "large" set of operators), things get more interesting, as now we are in a position to examine this using 2-modular representations, or equivalently, the $2S_n$-modules $2^{n\times n}$.

In the case of a single unary operator, our algebra is fairly simple, and for a set of k-ary operators, we will have to resort to tensor powers of the binary matrix algebra. For now, let's consider a unary operator $f : 2^n \to 2^n$, f's values are determined by its values on the atoms, by complete distributivity and atomicity, so we are interested in the function space $(2^n)^{\{n\}}$, and will heretofore reagard operators of functions $f : \{n\} \to 2^n$.

1.1 Unary Operator Algebra

We can easily bijectively associate the binary matrix algebra $2^{n\times n}$ with the set of unary operators on 2^n: Indeed, if an arbitrary matrix entry b_{ij} in a matrix in the algebra is 1, we can interpret that as expressing the fact that atom i is in the atomic decomposition (join of all atoms below it) of the Boolean element that atom j maps to under the matrix's corresponding given operator. A 0, on the other hand, would signify that the atom i is not in atom j's image's decomposition. So, $a_{ij} := \chi_{\{k\in\{n\}:k\leq f(j)\}}(i)$, where χ is the characteristic function. Note

that the ith row of a binary matrix $A = (a_{ij})$ is the set of all atoms whose image under $f : \{n\} \to 2^n$ dominate i. More formally, $a_{i\{n\}} := (\chi_{\{k\in\{n\}:i\leq f(k)\}}(\{n\})$ and $a_{\{n\}j} := (\chi_{\{k\in\{n\}:k\leq f(j)\}}(\{n\})$. (This suggests always associating a "transpose operator", $f^T : \{n\} \to 2^n$ defined to correspond with $a^T_{ij} := a_{ji}$ that maps each given atom, once again, to the atoms whose images are above it).

And for the other direction, we can define a specific operator corresponding to a binary matrix to be mere matrix multiplication by a Boolean element's decomposition vector, which we see corresponds to a completely distributive, normal Boolean operator by construction, needing only to define the operator on the "standard basis" $e_i := \delta_{i\{n\}}$, where i is an atom. (And mapping 0 arguments to 0.)

Having defined the bijective correspondence $\psi : \mathbf{Op}_{1,n} \to 2^{n\times n}$, where $\mathbf{Op}_{1,n}$ is the set of all unary operators (normal completely distributive operations), let's turn ψ into a 2-algebra isomorphism after we turn $\mathbf{Op}_{1,n}$ into a 2-algebra.

For the abelian addition on $\mathbf{Op}_{1,n}$, we see that pointwise exclusive-join is the perfect choice, since it gives each element an order of 2 to correspond with the matrix algebra's, thereby also defining additive inverses. Not to mention that it is consistent with binary matrix addition, and the standard binary fields, whereby the operator associated with the sum of two matrices maps a given atom (in column j) to the "1 entries" in j. which sum must have exactly one of the two operands' entries a 1, corresponding to exclusive join.

Formally :

$$(f + g)(x) := f(x) + g(x)\cdot -(f(x)\cdot g(x))$$

for $f, g \in \mathbf{Op}_{1,n}$, where the addition and multiplication on the right are performed in the Boolean algebra. This additive group is obviously abelian.

Next, we must define a multiplication of operators, and intuition tells us that it will involve meet, since that is the case with Boolean fields, but we must also involve the transpose opreator: $(f\cdot g)(j) := f^T(\{n\})\cdot g(j) := (f^T(i)\cdot g(j))_{1\leq i\leq n}$. To see this, let's associate the matrix F with f and G with g, so that the j-th column $F_{\{n\}j} := f(j)$.

$$\begin{aligned}(F \times G)_{ij} :&= F_{i\{n\}}\cdot G_{\{n\}j}\\ &= F^T_{\{n\}i}\cdot G_{\{n\}j}\\ &= f^T(i)\cdot g(j)\end{aligned} \tag{1}$$

So, we have finished defining the unary Boolean operator algebra operations, and we see that it's isomorphic to $2^{n\times n}$ by construction, so is certainly a 2-algebra.

1.2 Binary Operator Algebra

A binary operator $f : n^2 \to 2^n$ on the Cartesian square of n, intuitively seems describable with a tensor power. To represent $f(j, k)$ as a matrix, or in light of the previous section, to start with a tensor product of 2 $n \times n$ matrices A, B, we know $(A \otimes B) := (A_{ij}B)$, the scalar multiples of B by entries in A

(which can only be 0 or 1, of course). So, to define a corresponding binary operator, we merely define the image of (j, k) to be the list whose i-th entry is a 1 if $A_{ij} = B_{ik} = 1$, and 0 otherwise. And conversely and preliminarily, with every binary operator, we can associate a unary operator, whose value at the first argument is 1, which we can derive by complete distributivity. That function $f_1^1 : n \to 2^{n\times n}, j \mapsto f(1, j)$ will serve as the second operand in our tensor product. The first will be constructed accordingly : . . . . A bit superstitutious, but process of elimination?? hm?? at this level?? Will check to see if the above is valid, haha!

More to come.

2 Representations

Now, to return to the basic model of a finite Boolean algebra 2^n with normal, completely distributive operators $f_\xi, \xi \in \Xi$, in light of the 2-algebra isomorphism $\psi : \mathbf{Op}_{1,n} \to 2^{n\times n}$ above, we can examine the linear action of S_n on the vector space $2^{n\times n}$, and we see that this gives rise to the *defining Boolean operator representation*, $\delta : S_n \to GL(2^{n\times n})$ (afforded by the $2S_n$-module $2^{n\times n}$). It's clear that δ merely permutes the columns of a binary matrix. And of course we have the *defining Boolean representation*, similarly defined, corresponding to the action of a permutation of atoms on its Boolean sum: $\rho : S_n \to GL(2^n)$ (afforded by the $2S_n$-module 2^n). That is, it's a permutation representation on the binary vector space.

It turns out that the two homomorphisms are inherently connected by the formula $\delta(\sigma)(R)v = R\rho(\sigma)(v)$, for $R \in 2^{n\times n}$, $v \in 2^n$, and $\sigma \in S_n$. (We can write these more clearly as actions, $(\sigma\cdot R)v = R(\sigma\cdot v)$. Indeed this follows from the fact that multiplying a Boolean vector by a matrix whose columns have been permuted by σ, is the same as multiplying the non-permuted matrix by the vector whose elements have been permuted by σ. Note, the above equality is **not** the same as $\sigma\cdot(Rv) = \Sigma_{i\leq Rv}\sigma(i)$, the sum of the atoms obtained by permuting those below the Boolean vector Rv with σ.

3 Stabilizers

Given our defining unary Boolean operator representation, $\delta : S_n \to GL(2^{n\times n})$, a natural question to ask in its own right (which becomes more manageable in light of the operator algebra isomorphism) is which permutations stabilize (permute) sets of operators. In the matrix algebra formulation, this becomes: given a set $S \subset 2^{n\times n}$ of matrices, compute the stabilizer $\mathbf{Stab}_\delta(S) := \{\sigma \in S_n | \sigma S \subseteq S\}$.

So we lose nothing in merely restricting ourselves to the matrix representations in our quest for stabilizers. As one might expect, sets as important as these stabilizers don't come easy, in light of the current non-existence of polynomial time algorithsm to compute setwise stabilizers.

I'm in the process of trying to see if there's an elegant calculation of the automorphism group of $(2^n, S)$, in terms of perhaps the stabilizer $A := \mathbf{Stab}_\delta(S)$. It seems that they should be connected, and so far I reasoned that $\mathbf{Aut}(2^n, S) = \{\sigma \in S_n | \exists \eta \in A, \forall M \in S, x \in 2^n, M(\sigma \cdot x) = \sigma((\eta M) \cdot x)\}$ I'll let you know how the calculation turns out! Before, I'll get to the "music theory," I'll leave you with a

Conjecture 1. *For all $n \in \mathbb{N}$, there exists an $S \subset 2^{n \times n}$, such that the defining unary Boolean operator representation representation $\delta : S_n \to GL(2^{n \times n})$, when restricted to $S' := \boldsymbol{Stab}_{2^{n \times n}}(S)$, the setwise stabilizer of S, is equivalent to the restriction of the defining unary Boolean permutation representation $\rho : S_n \to GL(2^n)$ to S'.*

The reason I believe this might be true has to do with the fact that digital Boolean algebras have their "brains" embedded in their Boolean existence, and therefore they both register and manifest automorphisms in the same digital space.

4 Music Theory

We are not interested in a completist account of music theory which explains all of its mysterious combinatorics or neurological phenomena, or tries to create music algorithmically, because that is foreign to our purposes. We need the bare minimum of music theory required to aid in computation.

4.1 New Partial Approach

It's clear that since we're interested in the connections between a group (the setwise stabilizers in the defining representations) and certain strings of notes, that we should be interested in the free semigroup on 13 generators (12 for each note and one signifying the rests/sustains that provide the rhythm). Octave ambiguities wouldn't matter in the sense that all musical passges would still be musical with notes of the same value but different octave. It might strike one as less catchy than another, but it would undeniably be music, and the mere existence of a certain aesthetic best among equivalents is enough to earmark the equivalence class, even if not define a canonical representative.

So, the idea is that we will employ a partial algebra whose universe is the free semigroup on 13 generators, which we'll denote by F_{13}, and whose function symbols are any labels of the "build-up / resolution" pattern that is central to musical sensation.

The "built-up passages" (which of course can be nested), correspond to word arguments in a partial function, whose resolution is denoted by the function value word. By earmarking the parts through an audio device and a human listener, we can assemble a set of partial functions corresponding to the sensory flavor of the passage. Of course, a passage grows, and peformers usually like to

add to songs, as do composers and improvisers, so this just corresponds to new partial functions on the words (subwords of the musical passage).

The advantages of partial algebras is that they can construct "relatively free algebras over partial relative substructures" and we can already see some proof of this in connection with partial subalgebras. These come in three flavors – closed, relative, and weak relative. The closed subalgebra would correspond to some notion of "algorithmic melody construction" which as some algorithmic composers have found, can be used to undeniably create music. The relative subalgebra would correspond to restricting against certain examples of music that always sound bad to you. And the weak relative would correspond to filtering by removing notes and perhpas preserving rhythm.

This is more intuitive, and there are subtle points that have to be worked out, to both solidify the ideas, and so that we dont' lose sight of the forest for the trees.

More to come.

4.2 Old Partial Approach

As an alternative to the above beginnings, let's try using the technique of partial algebras in conjunction with Boolean algebras with operators, to utilize an interaction between the possible members of 13^L, where L is the length of our musical passage. We denote the set of all elements $\{1, \ldots, L\}$ by L_1. To reflect the structure in an improvised (or composed) passage, it's nice to use partial functions to express the possibilities and constraints that go through a musicians head as he improvises. For instance, a musician might know the last two notes he wants to play, and the first two notes, he wants to play, but not the middle two notes. This can be represented informally as, $t_1 \mapsto t_2 \to 13 \to 13 \mapsto t_3 \mapsto t_4$. In general, for any subset $S \subset L_1$, we can imagine an $L-$ bit-vector of the mask of the set, as it shifts throughout all the $L - |b_S|$ possibilities, where b_S is the number of spaces between the left-most 1 and the right-most 1. So, each partial function in the algebra that we will soon formally define represents, a musical choice, within the context of preceding notes, with or without gaps. So, the first 4 notes in the musical passage p, and if we know thisnote is t_5, we notate this by $t_1 \mapsto t_2 \mapsto t_3 \mapsto t_4 \mapsto t_5$. So, when we wish to examine all choices corresponding to this pattern of 4 consecutive notes, then we simply use an operation of arity 4, which is in $13^4 \to 13$. But, we are implicitly assuming that this corresponds to 4 consecutive notes. As we noted above, there can be very real musical choices made with gaps in the middle, so to remedy that we can using the power set algebra 2^L to index partial dictive functions So, the value $\{t_i\}_4$,

With this basic idea, we can examine the partial algebra $\mathscr{D}() := \{f_S | S \subset L_1\} where List helength of p$

A brief outline :

The letter $\mathscr{D}$ was chosen to denote diction.

More to come!

4.3 Old Flawed Approach

So, we start by looking at the Cartesian product of 13^M, where M is the (presumably large) number of notes in the musical passage, and 0-11 signify A-F#, while 12 signifies a rest or sustain. The idea is to get the bare minimal notion of music that could drive our computations. We don't distinguish between notes an octave apart, and we use the blank note 12 to define rhythm, which of course is quite complicated for complicated rhythms, but assuming nothing too avant-garde, we can capture most hummable melodies (in an equivalence class whose octave ambiguities are probably still musical).

We define the projection function $i_p : \{M\} \to 13$ which maps the index of the passage p to its note value. For instance, if the 3rd note in passage p is a C#, then $i_p(3) = 4$.

To create the BA with ops, we will use the standard procedure of starting with a relational structure, and then defining its uniquely determined complex algebra.

We'll let R_p denote a collection of relations of unrestricted arities determined by the passage p (and perhaps determined by music-theoretic principles). The primary relations that are uniquely determined by p are the unary relations $\rho_t \subset 13^M$, where $\rho_t a \leftrightarrow i(a) = t$. One useful class of operators is the k-ary relations $\sigma_{k,t} \subset 13^K$ where $(t_1, \ldots, t_k) \in \sigma_{k,t}$ iff there is a successive sequence of k components in p which contain t. We can also, instead of specifying only one tone to search for, search for a potential sub-passage. In any event, we will restrict our attention to the complex algebra B_p of R_p, which is the power set algebra on $\{M\}$, with an operator $f_\rho : \{M\}^k \to M$, for some relation $\rho \subseteq \{M\}^{k+1}$, sticking to the convention that

$$(t_1, \cdots, t_{k+1}) \in \rho \leftrightarrow f_\rho(t_1, \cdots, t_k) = t_{k+1}$$

.

So, the hope is that since Boolean homomorphisms are tantamount to ideals, which in finite BAs are simple enough, the main work will be generating enough relevant operators in B_p that are compatible with the defining Boolean operator matrix representation of S_N (which it'll be helpful to calculate explicitly soon!!) and a given set of operators S whose setwise stabilizer you want to find in the given BA $B := 2^N$, which imply the existence hopefully of a homomorphism of BAs with ops between B_p and B, whose stabilizers under the actions of the symmetric groups are comparable.

The hope is that it'll be much easier to compute setwise stabilizers in B_p than in B, since we get to choose our operators that are generated by the primary ones, and by whatever aesthetic constraints or persistent musical patterns we can describe. The variety is much more manageable in a music theoretic BA than in a BA with arbitrary operators.