(Re)constructing Code Loops

Abstract. The Parker loop is a central extension of the extended binary Golay code. It the prototypical example of a general class of nonassociative structures known as *code loops*, which have been studied from a number of different algebraic and combinatorial perspectives. This expository article aims also to highlight an experimental approach to computing in code loops, by a combination of a small amount of precomputed information and making use of the rich identities that code loops' twisted cocycles satisfy. As a byproduct we demonstrate that it is possible one can reconstruct the multiplication in the Parker loop from a mere fragment of its twisted cocycle, and we have found relatively large subspaces of the Golay code over which the Parker loop splits as a direct product.

1. INTRODUCTION. Associativity is one of the standard axioms for groups, along with inverses and the identity element. At first it can be difficult to imagine what algebraic structures could be like that are *not* associative. However, we are all familiar with the nonassociative operation of exponentiation: $(2^3)^2 \neq 2^{(3^2)}$, for example. However, exponentiation is an ill-behaved binary operation, with neither (a two-sided) inverse nor identity element, hence very much unlike a group. One class of nonassociative structures as close to being a group as possible are *Moufang loops*, which are more or less "groups without full associativity", having an identity element and two-sided inverses. This article is concerned with a special class of Moufang loops now known as *code loops*.

The first published example of a code loop appeared as a step in John Conway's construction of the Monster sporadic simple group [**4**]. That loop originally appeared in an unpublished manuscript by Richard Parker. A general study of loops of this type was then later made by Robert Griess [**7**], who gave them their name, as they can be built from doubly-even binary codes. For example, the Parker loop $\mathcal{P}$ is then constructed by starting with the famous extended Golay code. Griess also proved the existence of code loops by an algorithmic construction, which we adopt below. (More recent approaches will be discussed in section 6.)

The data required to describe a code loop is a function $\theta\colon \mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$, where $\mathbb{F}_2 = \{0, 1\}$ is the field with two elements, and $\mathcal{C} \subset (\mathbb{F}_2)^n$ is a doubly-even binary code; a subspace with a particular property that we recall below. The function θ must then satisfy a number of identities that use this property. For the sake of terminology, we refer to the elements of $\mathbb{F}_2$ as *bits*, and θ is called a *twisted cocycle*. Note, however, that both Conway's and Greiss's treatment of the Parker loop is rather implicit, merely using the properties of θ, not examining its structure. The point of the present paper is to give a concrete and efficient approach to code loops in general, and the Parker loop in particular—the latter especially so, given its use in constructing the Monster group.

Recall that the elements (or *words*) in a code $\mathcal{C}$, being vectors, can be combined by addition—this is a group operation and hence associative. The elements of a code loop consist of a pair: a code word and one extra bit. The extra

bit, together with θ, *twists* the addition so that the binary operation in the code loop is a *nonassociative operation*: $(x \star y) \star z \neq x \star (y \star z)$ in general.

While addition of words in a code is performed by coordinate-wise addition in $\mathbb{F}_2$ (that is, bitwise XOR), the algebraic operation in a code loop is not easily described in such an explicit way—unless one has complete knowledge of the function θ. It is the computation and presentation of such twisted cocycles that will mainly concern us in this article, using Griess's method [**7**, proof of Theorem 10]. As a result, we will observe some curious features of the Parker loop, obtained via experimentation and, it seems, previously unknown.

The function θ can be thought of as a table of bits, or even as a square image built from black and white pixels, with rows and columns indexed by elements of $\mathcal{C}$. One of the main results given here is that θ can be reconstructed from a much smaller subset of its values. If $\dim \mathcal{C} = 2k$, then while θ is a table of $(2^{2k})^2$ values, we need only store $\approx 2^{2k}$ values, and the rest can be reconstructed from equation (6) in Lemma 2 below. For the Parker loop this means that instead of storing a multiplication table with approximately 67 million entries, one only need store a 128×128 table of bits. This table, given in Figure 3, exhibits several structural regularities that simplify matters further.

We begin the article with a treatment that (re)constructs a small nonabelian finite *group* using a cocycle on a 2-dimensional $\mathbb{F}_2$-vector space. This will exhibit the technique we will use to construct code loops, in the setting of undergraduate algebra.

2. EXTENSIONS AND COCYCLES. Recall that the *quaternion group* Q_8, usually introduced in a first course in group theory, is the multiplicative group consisting of the positive and negative basis quaternions:

$$Q_8 = \{1, i, j, k, -1, -i, -j, -k\}.$$

The elements of Q_8 satisfy the identities

$$i^2 = j^2 = k^2 = -1, \quad ij = k.$$

There is a surjective group homomorphism $\pi\colon Q_8 \to \mathbb{F}_2 \times \mathbb{F}_2 =: V_4$ (the Klein 4-group), sending i to $(1,0)$ and j to $(0,1)$, and the kernel of π is the subgroup $(\{1,-1\}, \times) \simeq (\mathbb{F}_2, +)$. Moreover, this kernel is the *center* of Q_8, the set of all elements that commute with every other element of the group. This makes Q_8 an example of a *central extension*: $\mathbb{F}_2 \to Q_8 \to V_4$.

Now Q_8 is a nonabelian group, but both $\mathbb{F}_2$ and V_4 are abelian. One might think that it shouldn't be possible to reconstruct Q_8 from the latter two groups, but it is! That is, if we are given some extra information that uses only the two abelian groups. There is a function $s\colon V_4 \to Q_8$, called a *section*, defined by

$$(0,0) \mapsto 1 \qquad (1,0) \mapsto i \qquad (0,1) \mapsto j \qquad (1,1) \mapsto k.$$

This *almost* looks like a group homomorphism, but it is not, as $(1,0) + (1,0) = (0,0)$ in V, but $s(1,0)s(1,0) = i^2 \neq 1 = s(0,0)$ in Q_8. We can actually measure the failure of s to be a group homomorphism by considering the two-variable function

$$d\colon V_4 \times V_4 \to \mathbb{F}_2$$

defined by $(-1)^{d(v,w)} = s(v)s(w)s(v+w)^{-1}$. This function is called a *cocycle*, or more properly a 2-cocycle (for a treatment of the general theory, see e.g. [**1**], and Chapter IV in particular). It is a nice exercise to see that $s(v)s(w)s(v+w)^{-1}$ is always ± 1, so that this definition makes sense. The values of $d(v,w)$ are given as

$v \setminus w$	00	10	01	11
00	0	0	0	0
10	0	1	1	0
01	0	0	1	1
11	0	1	0	1

where we write 00 for $(0,0)$, 10 for $(1,0)$, and so on. Clearly, if s *were* a homomorphism, d would be the zero function. One can check that d satisfies the *cocycle identities*

$$d(v,w) - d(u+v,w) + d(u,v+w) - d(u,v) = 0 \tag{1}$$

for all triples $u, v, w \in V_4$. It is also immediate from the definition that $d(0,0) = 0$. An alternative visualization is given in Figure 1.

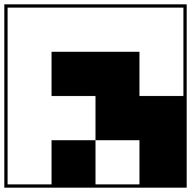

Figure 1. A 4×4 array giving the values of the cocycle $d\colon V_4 \times V_4 \to \mathbb{F}_2$, with white $= 0$, black $= 1$. The order of the row/column labels is 00, 10, 01, 11.

The reason for this somewhat mysterious construction is that we can build a bijection of *sets* using s and the group isomorphism $\mathbb{F}_2 \simeq \{1, -1\}$, namely the composite

$$\phi\colon \mathbb{F}_2 \times V_4 \simeq (\{1\} \times V_4) \cup (\{-1\} \times V_4) \longrightarrow \{1,\, i,\, j,\, k\} \cup \{-1,\, -i,\, -j,\, -k\} = Q_8\,.$$

Now, if we define a new product operation using the cocycle d on the underlying set of $\mathbb{F}_2 \times V_4$ by

$$(s,v) *_d (t,w) := (s + t + d(v,w), v + w),$$

then the cocycle identities ensure that this is in fact associative and further, a group operation. Finally, ϕ can be checked to be a homomorphism for multiplication in Q_8 and for $*_d$, hence is a group isomorphism.

Thus we can reconstruct, at least up to isomorphism, the nonabelian group Q_8 from the two abelian groups V_4 and $\mathbb{F}_2$, together with the cocycle

$$d\colon V_4 \times V_4 \to \mathbb{F}_2.$$

If we didn't know about the group structure of Q_8 already, we could construct it from scratch using d. We can also build loops (particularly code loops) in the same way, which we will now outline.

3. TWISTED COCYCLES AND LOOPS. The construction in the previous section is a fairly typical case of reconstructing a central extension from a cocycle (although in general one does not even need the analog of the group V_4 to be abelian). However, we wish to go one step further, and construct a structure with a *nonassociative* product from a pair of abelian groups: the group $\mathbb{F}_2$ and additive group of a vector space V over $\mathbb{F}_2$. Instead of a cocycle, we use a *twisted* cocycle: a function $\alpha\colon V \times V \to \mathbb{F}_2$ like d that, instead of (1), satisfies

$$\alpha(v,w) - \alpha(u+v,w) + \alpha(u,v+w) - \alpha(u,v) = f(u,v,w)$$

for a nonzero *twisting function* $f\colon V \times V \times V \to \mathbb{F}_2$. We will assume that α satisfies $\alpha(\underline{0},v) = \alpha(v,\underline{0}) = 0$ for all $v \in V$, a property that holds for the cocycle d in the previous section. The 'twisted cocycle' terminology comes from geometry, where they appear in other guises; another term used is *factor set*.

From a twisted cocycle α the set $\mathbb{F}_2 \times V$ can be given a binary operation $*_\alpha$:

$$(s,v) *_\alpha (t,w) := (s+t+\alpha(v,w), v+w). \tag{2}$$

We denote the product $\mathbb{F}_2 \times V$ of sets equipped with this binary operation by $\mathbb{F}_2 \times_\alpha V$. If the twisting function f is zero, then α is a cocycle, $*_\alpha$ is an associative operation, and $\mathbb{F}_2 \times_\alpha V$ is a group.

Definition. A *loop* is a set L with a binary operation $\star\colon L \times L \to L$, a unit element $e \in L$ such that $e \star x = x \star e = x$ for all $x \in L$, and such that for each $z \in L$, the functions $r_z(x) = x \star z$ and $\ell_z(x) = z \star x$ are bijections $L \to L$. A *homomorphism* of loops is a function preserving multiplication and the unit element.

Informally, this means that every element $z \in L$ has a left inverse and a right inverse for the operation $\star$, and these are unique—but may be different in general. The following is a worthwhile exercise using the twisting function and the assumption that $\alpha(\underline{0},v) = \alpha(v,\underline{0}) = 0$.

Lemma 1. *The operation $*_\alpha$ makes $\mathbb{F}_2 \times_\alpha V$ into a loop, with identity element $(0,\underline{0})$. There is a homomorphism $\pi\colon \mathbb{F}_2 \times_\alpha V \to V$ that projects onto the second factor, and whose kernel is $\mathbb{F}_2 \times \{\underline{0}\}$.*

Groups are examples of loops, but they are, in a sense, the uninteresting case. Arbitrary loops are quite badly behaved: even apart from the product being nonassociative, left and right inverses may not coincide, and associativity can fail for iterated products of a single element. But there is a special case introduced by Ruth Moufang [**11**], with better algebraic properties, while still permitting nonassociativity.

Definition. A *Moufang loop* is a loop $(L,\star)$ satisfying the identity

$$x \star (y \star (x \star z)) = ((x \star y) \star x) \star z$$

for all choices of elements $x, y, z \in L$.

The most famous example of a Moufang loop is probably the set of nonzero octonions under multiplication, although there are many less-known finite examples. A key property of a Moufang loop L is that any subloop $\langle x,y\rangle < L$ generated by a pair of elements x, y is in fact a group. By a *subloop* of

L we mean a subset containing e that is closed under the operation $\star$. As a corollary, powers of a *single* element are well-defined, and do not require extra bracketing: $x \star (x \star x) = (x \star x) \star x =: x^3$, for example. Additionally, the left and right inverses always agree in a Moufang loop, so that for each $x \in L$, there is a unique x^{-1} such that $x \star x^{-1} = x^{-1} \star x = e$. Code loops, defined below as a special case of the construction of $\mathbb{F} \times_\alpha V$, are examples of Moufang loops.

Example 1. Let $V = (\mathbb{F}_2)^3$. The 16-element Moufang loop $M := M_{16}(C_2 \times C_4)$ of [**2**, Theorem 2] is isomorphic to $\mathbb{F}_2 \times_\mu V$, where $\mu\colon V \times V \to \mathbb{F}_2$ is the twisted cocycle given by the 8×8 array of bits in Figure 2.

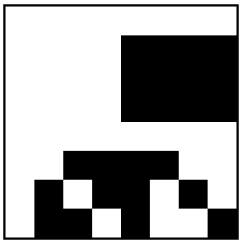

Figure 2. Twisted cocycle for the Moufang loop M of Example 1; white = 0, black = 1. The order of the row/column labels is 000, 100, 010, 110, 001, 101, 011, 111.

Notice that the first four columns/rows correspond to the subgroup $U \subset V$ generated by 100 and 010, and that the restriction of μ to $U \times U$ is identically zero (i.e., white).

This means that the *restriction* $M\big|_U < M$—the subloop of M whose elements are mapped to U by $M \simeq \mathbb{F}_2 \times_\mu V \to V$—is isomorphic to the direct product $\mathbb{F}_2 \times U$, using the definition (2) of the product, and in particular is a group.

4. CODES AND CODE LOOPS. To describe the twisting function f for our code loops, we need to know about some extra operations that exist on vector spaces over the field $\mathbb{F}_2$. For W an n-dimensional vector space over $\mathbb{F}_2$ and vectors $v, w \in W$, there is a new vector $v \,\&\, w \in W$ given by

$$v \,\&\, w := (v_1 w_1,\, v_2 w_2,\, \ldots,\, v_n w_n).$$

If we think of such vectors as binary strings, then this operation is bitwise AND (hence the notation). Note that if we take a code $\mathcal{C} \subset (\mathbb{F}_2)^n$, then $\mathcal{C}$ is not guaranteed to be closed under this operation. The other operation takes a vector $v \in W$ and returns its *weight*: the sum, as an integer, of its entries: $|v| := v_1 + \cdots + v_n$. Equivalently, it is the number of nonzero entries in v.

The twisting function for a code loop is then a combination of these two, namely $f(u, v, w) := |u \,\&\, v \,\&\, w|$. However, as alluded to above, we are going to ask that further identities hold. For these identities to make sense we need to start with a code with the special property of being *doubly even*.

Definition. A code $\mathcal{C} \subset (\mathbb{F}_2)^n$ is *doubly even* if for every word $v \in \mathcal{C}$, $|v|$ is divisible by 4.

Example 2. The *Hamming (8,4) code* is the subspace $\mathcal{H} \subset (\mathbb{F}_2)^8$ spanned by the four row vectors

$$\begin{array}{c} 10000111 \\ 01001011 \\ 00101101 \\ 00011110 \end{array}$$

and is doubly even.

A more substantial example is given by the Golay code.

Example 3. The (extended binary) Golay code $\mathcal{G} \subset (\mathbb{F}_2)^{24}$ is the span of the following (row) vectors, denoted $b_1, \ldots, b_6$ (left) and $b_7, \ldots, b_{12}$ (right):

000110000000010110100011	101001011100111001111111
101001111101101111110001	100000011100001001001100
000100000000100100111110	000001000000111001001110
010000000010000110101101	100000001000111000111000
000000000010010101010111	100000000100101000010111
100000000000100111110001	011011000001111011111111

This basis is different from the more usual ones (e.g., [**5**, Figure 3.4]), which can be taken as the rows of a 12×24 matrix whose left half is the 12×12 identity matrix. Our basis, however, allows us to demonstrate some interesting properties.

The inclusion/exclusion formula applied to counting nonzero entries allows us to show that, for all v and w in any doubly even code $\mathcal{C}$,

$$|v + w| + |v \,\&\, w| = |v| + |w| - |v \,\&\, w|\,.$$

In other words: $|v \,\&\, w| = \frac{1}{2}(|v| + |w| - |v + w|)$, which implies that $|v \,\&\, w|$ is divisible by 2. Thus for words v, w in a doubly even code, both $\frac{1}{4}|v|$ and $\frac{1}{2}|v \,\&\, w|$ are integers.

Definition (Griess [7]). Let $\mathcal{C}$ be a doubly even code. A *code cocycle*[1] is a function $\theta\colon \mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$ satisfying the identities

$$\theta(v, w) - \theta(u + v, w) + \theta(u, v + w) - \theta(u, v) = |u \,\&\, v \,\&\, w| \pmod 2; \tag{3}$$

$$\theta(v, w) + \theta(w, v) = \tfrac{1}{2}|v \,\&\, w| \pmod 2; \tag{4}$$

$$\theta(v, v) = \tfrac{1}{4}|v| \pmod 2. \tag{5}$$

A *code loop* is then a loop arising as $\mathbb{F}_2 \times_\theta \mathcal{C}$ (up to isomorphism) for some code cocycle θ.

Example 4. Define $\mathcal{C} := \operatorname{span}\{b_{10}, b_{11}, b_{12}\}$, using the basis vectors from Example 3. Then the composite function $\mathcal{C} \times \mathcal{C} \simeq (\mathbb{F}_2)^3 \times (\mathbb{F}_2)^3 \xrightarrow{\mu} \mathbb{F}_2$, using the twisted cocycle μ in Figure 2, is a code cocycle. This makes the loop M from Example 1 a code loop.

There is a notion of what it means for two twisted cocycles to be equivalent, and equivalent twisted cocycles give isomorphic loops. As part of [**7**, Theorem 10], Griess proves that all code cocycles for a given doubly even code are equivalent, and hence give isomorphic code loops.

[1] Griess uses the alternative term "factor set".

5. GRIESS'S ALGORITHM AND ITS OUTPUT. The algorithm that Griess describes in the proof of [**7**, Theorem 10] to construct code cocycles for a code $\mathcal{C}$ takes as input an ordered basis $\{b_0, \ldots, b_{k-1}\}$ for $\mathcal{C}$. A code cocycle is then built up inductively over larger and larger subspaces $V_i = \operatorname{span}\{b_1, \ldots, b_i\}$.

However, the description by Griess is more of an outline, using steps like "determine the cocycle on such-and-such subset using identity X," where X refers to one of (3), (4), (5), or corollaries of these. We have reconstructed the process in detail in Algorithm 1.

We implemented Algorithm 1 in the language Go [**12**], together with diagnostic tests, for instance to verify the Moufang property. The output of the algorithm is a code cocycle, encoded as a matrix of ones and zeros, and can be displayed as an array of black and white pixels. There are steps where an arbitrary choice of a single bit is allowed; we consistently took this bit to be 0. For the Golay code this image consists of slightly over 16 million pixels.

As a combinatorial object, a code cocycle $\theta\colon \mathcal{G} \times \mathcal{G} \to \mathbb{F}_2$ constructed from Algorithm 1 using the basis in Example 3 can be too large and unwieldy to examine for any interesting structure. Moreover, to calculate with the Parker loop $\mathcal{P} := \mathbb{F}_2 \times_\theta \mathcal{G}$ one needs to know all 16 million or so values of θ. It is thus desirable to have a method that will calculate values of θ by a method shorter than Algorithm 1.

Lemma 2. *Let $\mathcal{C}$ be a doubly even code and θ a code cocycle on it. Given $\mathcal{C} = V \oplus W$ a decomposition into complementary subspaces, then for $v_1, v_2 \in V$ and $w_1, w_2 \in W$,*

$$\begin{aligned}\theta(v_1 + w_1, v_2 + w_2) = {} & \theta(v_1, v_2) + \theta(w_1, w_2) + \theta(v_1, w_1) \\ & + \theta(w_2, v_2) + \theta(v_1 + v_2, w_1 + w_2) \\ & + \tfrac{1}{2}|v_2 \,\&\, (w_1 + w_2)| + |v_1 \,\&\, v_2 \,\&\, (w_1 + w_2)| \\ & + |w_1 \,\&\, w_2 \,\&\, v_2| + |v_1 \,\&\, w_1 \,\&\, (v_2 + w_2)| \pmod 2 .\end{aligned} \tag{6}$$

Proof. We apply the identity (3) three times and then the identity (4) once:

$$\begin{aligned}\theta(v_1 + w_1, v_2 + w_2) &= \theta(v_1, w_1) + \theta(v_1, v_2 + w_1 + w_2) + \theta(w_1, v_2 + w_2) \\ &\quad + |v_1 \,\&\, w_1 \,\&\, (v_2 + w_2)| \\ &= \theta(v_1, w_1) + \big\{\theta(v_1, v_2) + \theta(v_1 + v_2, w_1 + w_2)) \\ &\quad + \theta(v_2, w_1 + w_2) + |v_1 \,\&\, v_2 \,\&\, (w_1 + w_2)|\big\} \\ &\quad + \big\{\theta(w_1, w_2) + \theta(w_1 + w_2, v_2) + \theta(w_2, v_2) + |w_1 \,\&\, w_2 \,\&\, v_2|\big\} \\ &\quad + |v_1 \,\&\, w_1 \,\&\, (v_2 + w_2)| \\ &= \theta(v_1, w_1) + \big\{\theta(v_1, v_2) + \theta(v_1 + v_2, w_1 + w_2)) \\ &\quad + |v_1 \,\&\, v_2 \,\&\, (w_1 + w_2)|\big\} \\ &\quad + \big\{\theta(w_1, w_2) + \theta(w_2, v_2) + |w_1 \,\&\, w_2 \,\&\, v_2|\big\} \\ &\quad + |v_1 \,\&\, w_1 \,\&\, (v_2 + w_2)| + \tfrac{1}{2}|v_2 \,\&\, (w_1 + w_2)|. \end{aligned}$$

■

Observe that in Lemma 2, on the right-hand side of (6), the code cocycle θ is only ever evaluated on vectors from the subset $V \cup W \subset \mathcal{C}$. This means we

can throw away the rest of the array and still reconstruct arbitrary values of θ using (6). If we assume that $\mathcal{C}$ is $2k$-dimensional, and that V and W are both k-dimensional, then the domain of the restricted θ has $(2^k + 2^k - 1)^2 = 2^{2(k+1)} - 2^{k+2} + 1 = O((2^k)^2)$ elements. Compared to the full domain of θ, which has $2^{2k} \times 2^{2k} = (2^k)^4$ elements, this is roughly a square-root saving.

However, the rationale for choosing the subspaces V and W is that one is looking for a reduction in the *entropy* (roughly, the disorderedness) of the restricted code cocycle. This is true, even if the size of $V \cup W$ is not minimized by choosing V and W to have dimension $(\dim \mathcal{C})/2$ (or as close as possible).

And, now it should be clear why the Golay code basis in Example 3 was partitioned into two lists of six vectors: we can reconstruct all 16 777 216 values of the resulting code cycle θ, and hence the Parker loop multiplication, from a mere $2^{14} - 2^8 + 1 = 16\,129$ values. The span of the left column of vectors in Example 3 is the subspace $V \subset \mathcal{G}$, and the span of the right column of vectors is $W \subset \mathcal{G}$.

The top left quadrant of Figure 3 then contains the restriction of θ to $V \times V$, and the bottom right quadrant the restriction to $W \times W$. The off-diagonal quadrants contain the values of θ restricted to $V \times W$ and $W \times V$.

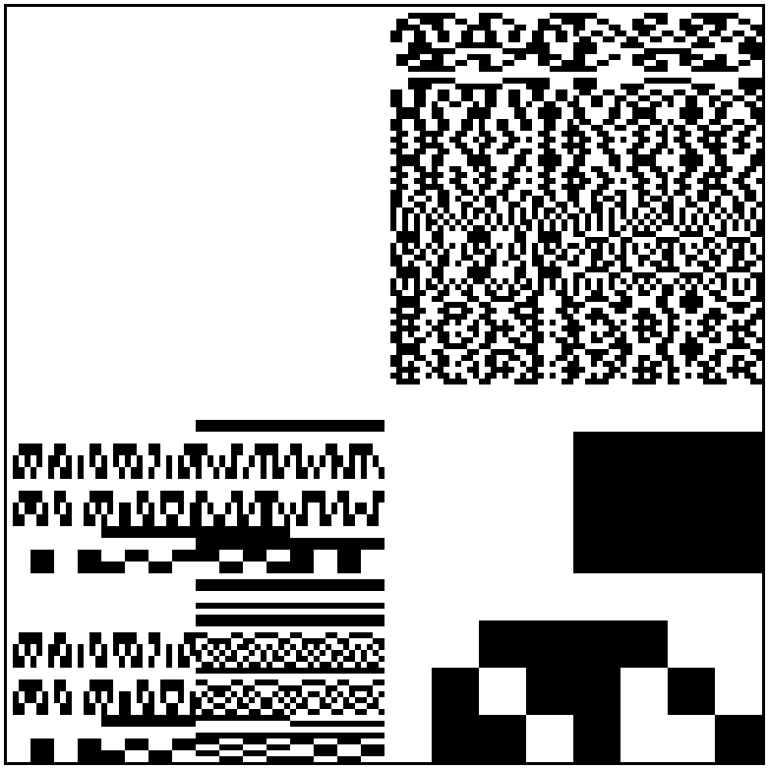

Figure 3. The restriction of the Parker loop code cocycle θ to $(V \cup W)^2$. A machine-readable version is available in [**12**] or as an ancillary file. The order of the row/column labels is $\underline{0}, b_1, b_2, b_1 + b_2, b_3, \ldots, b_1 + \cdots + b_6,\ b_7, b_8, b_7 + b_8, \ldots, b_7 + \cdots + b_{12}$.

From Figure 3 and (2) we can see that the restriction of the Parker loop $\mathcal{P}\big|_V$ is a direct product (hence is a group), as $\theta\big|_{V \times V}$ is identically zero. Moreover,

the restriction of $\mathcal{P}\big|_W$ is isomorphic to the direct product $(\mathbb{F}_2)^3 \times M$, where M is the Moufang loop from Example 1. This is because what was a single pixel in Figure 2 is now an 8×8 block of pixels in the lower right quadrant of Figure 3.

6. DISCUSSION AND COMPARISON. The Parker loop $\mathcal{P}$ is a sporadic object that has spawned a small industry trying to understand and construct code loops in general, as a class of Moufang loops that are fairly easy to describe.

Aside from the description by Griess using code cocycles, there was an unpublished description by Parker, an unpublished thesis [**9**], characterizations as loops with specified commutators and associators [**3**], an iterative construction using centrally twisted products [**8**], and a construction using groups with triality [**13**]. The LOOPS library [**14**] for software package GAP [**6**] contains all the code loops of order 64 and below, although "the package is intended primarily for quasigroups and loops of small order, say up to 1000." Even the more recent [**15**], which classifies code loops up to order 512 in order to add them to the LOOPS package, falls short of the Parker loop's 8192 elements; the authors say "our work suggests that it will be difficult to extend the classification of code loops beyond order 512." In principle, there is nothing stopping the construction of $\mathcal{P}$ in LOOPS, but it will essentially be stored as a multiplication table, which would comprise 67 108 864 entries, each of which is a 13-bit element label. The paper [**10**] describes an algebraic formula for a code cocycle that will build the Parker loop, as a combination of the recipe in the proof of Proposition 6.6 and the generating function in Proposition 9.2 appearing there. This formula is a polynomial with 330 cubic terms and 12 quadratic terms in 24 variables, being coefficients of basis vectors of $\mathcal{G}$. To compare, combined with the small amount of data in Figure 3 together with a labelling of rows/columns by words of the Golay code, Lemma 2 only requires eight terms, of which four are cubic and the rest come from the 16 129 stored values of θ (the term $\theta(v_1, v_2)$ vanishes identically, for $\mathcal{P}$). Since large-scale computation in large code loops requires optimising the binary operation, we have found a space/time trade-off that vastly improves on existing approaches.

In addition to computational savings, the ability to visually explore the structure of code loops during experimentation more generally is a novel advance—the recognition of $(\mathbb{F}_2)^3 \times M$ inside $\mathcal{P}$ was purely by inspection of the picture of the code cocyle then consulting the small list of Moufang loops of small order [**2**]. Discovery of the basis in Example 3 occurred by walking through the spaces of bases of subcodes and working with the heuristic that more regularity in the appearance of the code cocycle is better. Additionally, our software flagged when subloops thus considered were in fact associative, and hence a group, leading to the discovery of the relatively large elementary subgroups $(\mathbb{F}_2)^7 < \mathcal{P}$ and $(\mathbb{F}_2)^6 < (\mathbb{F}_2)^3 \times M < \mathcal{P}$.

Finally, one can also remark that because of the identity (4), one can replace the formula in Lemma 2 by one that is only ever evaluated on V^2, W^2 or $V \times W$, with just one more term. This allows the reconstruction the top right quadrant of Figure 3 from the bottom left quadrant. Thus one can describe the Parker loop as being generated by the subloops $(\mathbb{F}_2)^7$ and $(\mathbb{F}_2)^3 \times M$, with relations coming from the information contained in the bottom left quadrant of Figure 3, and the formulas (4) and (6). The regularity in that bottom left quadrant is intriguing, and perhaps indicative of further simplifications; this will be left to future work.

Data: Basis $B = \{b_0, b_1, \ldots, b_{k-1}\}$ for the code $\mathcal{C}$
Result: Code cocycle $\theta\colon \mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$, encoded as a square array of elements from $\mathbb{F}_2$, with rows and columns indexed by $\mathcal{C}$

```
// Initialise
forall c1, c2 ∈ C do
    θ(c1, c2) ← 0
end
θ(b0, b0) ← (1/4) |b0|

forall 1 ≤ i ≤ length(B) do
    Define Vi := span{b0, ..., bi−1}
    // (D1) define theta on {bi} x Vi then deduce on Vi x {bi}
    forall v ∈ Vi do
        if v ≠ 0 then
            θ(bi, v) ← random                          // In practice, random = 0
            θ(v, bi) ← (1/2) |v & bi| + θ(bi, v)
        else
            // θ(bi, v) is already set to 0
            θ(v, bi) ← (1/2) |v & bi|
        end
    end
    // (D2) deduce theta on {bi} x Wi and Wi x {bi}
    forall v ∈ Vi do
        θ(bi, bi + v) ← (1/4) |bi| + θ(bi, v)
        θ(bi + v, bi) ← (1/2) |bi & (bi + v)| + (1/4) |bi| + θ(bi, v)
    end
    // (D3) deduce theta on Wi x Wi
    forall v1 ∈ Vi do
        forall v2 ∈ Vi do
            w ← bi + v2
            a ← θ(v1, bi)
            b ← θ(v1, bi + w)
            c ← θ(w, bi)
            r ← (1/2) |v1 & w| + a + b + c
            θ(w, bi + v1) ← r
        end
    end
    // (D4) deduce theta on Wi x Vi and Vi x Wi
    forall v1 ∈ Vi do
        forall v2 ∈ Vi do
            w ← bi + v2
            a ← θ(w, v1 + w)
            θ(w, v1) ← (1/4) |w| + a
            θ(v1, w) ← (1/2) |v1 & w| + (1/4) |w| + a
        end
    end
end
```

Algorithm 1: Reverse-engineered from proof of [**7**, Theorem 10].

REFERENCES

1. Brown, K. (1994). *Cohomology of groups*, vol. 87 of *Graduate Texts in Mathematics*. Springer-Verlag, New York. Corrected reprint of the 1982 original.
2. Chein, O. (1974). Moufang loops of small order. I. *Trans. Amer. Math. Soc.*, 188: 31–51. http://dx.doi.org/10.2307/1996765.
3. Chein, O., Goodaire, E. G. (1990). Moufang loops with a unique nonidentity commutator (associator, square). *J. Algebra*, 130(2): 369–384. http://dx.doi.org/10.1016/0021-8693(90)90087-5.
4. Conway, J. H. (1985). A simple construction for the Fischer-Griess monster group. *Invent. Math.*, 79(3): 513–540. http://dx.doi.org/10.1007/BF01388521.
5. Conway, J. H., Sloane, N. J. A. (1999). *Sphere packings, lattices and groups*, vol. 290 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, New York, 3rd ed.
6. The GAP Group (2019). GAP – Groups, Algorithms, and Programming, Version 4.10.2. https://www.gap-system.org.
7. Griess, Jr., R. L. (1986). Code loops. *J. Algebra*, 100(1): 224–234. http://dx.doi.org/10.1016/0021-8693(86)90075-X.
8. Hsu, T. (2000). Explicit constructions of code loops as centrally twisted products. *Math. Proc. Cambridge Philos. Soc.*, 128(2): 223–232. http://dx.doi.org/10.1017/S030500419900403X.
9. Johnson, P. M. (1988). Gamma spaces and loops of nilpotence class two. Ph.D. thesis, University of Illinois at Chicago.
10. Morier-Genoud, S., Ovsienko, V. (2011). A series of algebras generalizing the octonions and Hurwitz-Radon identity. *Comm. Math. Phys.*, 306(1): 83–118. http://dx.doi.org/10.1007/s00220-011-1279-9.
11. Moufang, R. (1935). Zur Struktur von Alternativkörpern. *Math. Ann.*, 110(1): 416–430. http://dx.doi.org/10.1007/BF01448037.
12. library authors (2019). `codeloops` library. https://adelaide.figshare.com/s/e1cf101636659583a124
13. Nagy, G. P. (2008). Direct construction of code loops. *Discrete Math.*, 308(23): 5349–5357. http://dx.doi.org/10.1016/j.disc.2007.09.056.
14. Nagy, G., Vojtěchovský, P. (2018). LOOPS – a GAP package, Version 3.4.1. https://gap-packages.github.io/loops/.
15. O'Brien, E. A., Vojtěchovský, P. (2017). Code loops in dimension at most 8. *J. Algebra*, 473: 607–626. http://dx.doi.org/10.1016/j.jalgebra.2016.11.006.