BRAIDS AND JUGGLING PATTERNS

MATTHEW MACAULEY
MICHAEL ORRISON

Abstract

Suppose we juggle an n-ball siteswap pattern as we walk forward. The paths of the balls will trace out a braid in 3-space with n strings. A braid can be represented algebraically as an element of a braid group. Braid groups give us a way to study the topology of juggling patterns.

1 Prelimaries

1.1 The Braid Group

Definition 1.1 *Consider two planar parallel segments X and Y in $\mathbb{R}^3$ each containing n distinct points, $\{x_i\}$ and $\{y_i\}$. An n-braid is a collection of n curves $\{b_i\}$, where $b_i : [0,1] \longrightarrow \mathbb{R}^3$ for each b_i and the following conditions hold:*

1. *Each b_i has one endpoint at one of the x_i's and one endpoint a y_j.*
2. *All the b_i's are pairwise disjoint.*
3. *Every plane parallel to X and Y and normal to the plane containing them either intersects each b_i at exactly one point or is disjoint from all of them.*

The easiest way to draw a braid is to draw its projection onto a plane and denote which strand is on top at each crossing. For each braid, we can choose a projection such that no three strands meet at any one point, and any two strands intersect at a finite number of points. The first diagram in Figure 1 is a braid, but the second is not because it violates the third property.

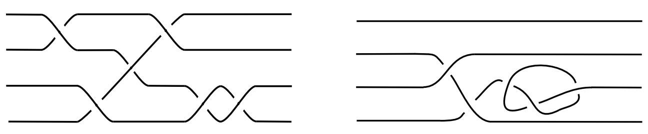


Figure 1: A braid on four strings, and an illegal braid.

This will be our conventional way of drawing braids. We can put an algebraic structure on the set of braids on n strings, or n-braids, with a finite number of crossings when projected onto a plane. Any braid can be generated by repeatedly crossing adjacent strings. Starting from one end of the braid and moving to the other, we can list all the crossings one at a time as given by the following rules: At any point, if the current ith strand from the bottom crosses under the $(i+1)$th strand, call it σ_i. If it crosses over, call it σ_i^{-1}. Figure 2 is an example of this. Any braid can be expressed as a word of the σ_i's and σ_i's.

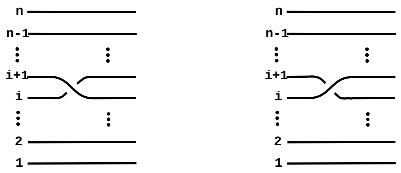


Figure 2: The ith generator of the braid group, σ_i, and its inverse.

Two braids are considered equivalent if they can be expressed by the same word. There are two relations that can be useful when determining whether two braids are equivalent. The first braid relation is $\sigma_i\sigma_j = \sigma_j\sigma_i$ iff $|i-j| \geq 2$. This is intuitive, because two crossings far enough apart can be moved horizontally independently as shown by the diagram in Figure 3.

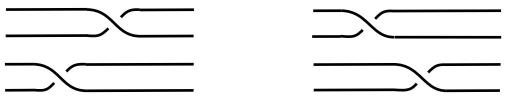


Figure 3: The first braid relation: $\sigma_i\sigma_j = \sigma_j\sigma_i$ iff $|i-j| \geq 2$.

The second braid relation is $\sigma_k\sigma_{k+1}\sigma_k = \sigma_{k+1}\sigma_k\sigma_{k+1}$. In knot theory, this is called the third Reidemeister move, which allows a strand to be moved past a crossing. Figure 4 gives an example of this relation.

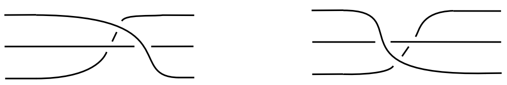


Figure 4: The second braid relation: $\sigma_k\sigma_{k+1}\sigma_k = \sigma_{k+1}\sigma_k\sigma_{k+1}$.

The set of all n-braids forms the *braid group*, and the two relations in fact generate the braid group. Thus the braid group on n strings, denoted $\mathbf{B}_n$, has presentation

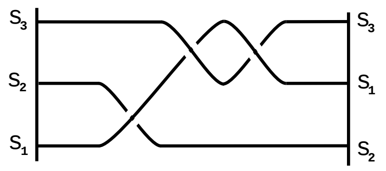
$$\mathbf{B}_n = \left\langle \sigma_1, \ldots \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j = \sigma_j\sigma_i \text{ iff } |i-j| \geq 2 & i,j \in \{1,\ldots,n-1\} \\ \sigma_k\sigma_{k+1}\sigma_k = \sigma_{k+1}\sigma_k\sigma_{k+1} & k \in \{1,\ldots,n-2\} \end{array} \right\rangle .$$

The proof that these two relations generate the braid group is quite involved and will not be given here. A proof can be found in Chapter 1, Section 3, of [2].

To each braid we can assign a permutation based on the order of the strings at the end of the braid. A braid is called a *pure braid* if its permutation is the identity. The identity of the braid group, the unbraid, is an example of a pure braid. The set of all pure braids on n strings, denoted $\mathbf{P}_n$, is a normal subgroup in $\mathbf{B}_n$ (see Chapter 1, Proposition 4.5 in [2]).

1.2 The Crossing Invariant

There is a simple but useful braid invariant for pure braids called the crossing number. If we number each string, then we can define $\mathrm{cr}(i,j)$ be the number of times the i^{th} string passes behind the j^{th} string from below, minus the number of times the i^{th} string passes behind the j^{th} string from above. The crossing number is a braid invariant for pure braids. Still, the crossing number will be a very useful tool later in the chapter. An example of the crossing number of a braid is given in Figure 5.



$$\begin{array}{ll} \mathrm{cr}(1,2)=0 & \mathrm{cr}(2,1)=-1 \\ \mathrm{cr}(1,3)=1 & \mathrm{cr}(3,1)=\ \ 1 \\ \mathrm{cr}(2,3)=0 & \mathrm{cr}(3,2)=\ \ 0 \end{array}$$

Figure 5: An example of the crossing numbers.

2 Braids of Juggling Patterns

If we want to examine the braids of juggling patterns we have to set a standard for the number of hands and the throwing and catching locations of the balls. Siteswap notation does not distinguish this, and varying this will change the flight paths of the balls and possibly the braid. We will start with a simple one-hand model. Balls are caught at a fixed location and throws can be made from either side.

The best way to analyze this braid is to construct it from a profile braid. This works nicely because we can determine the over/under crossings straight from the stack sequence. There are two types of throws that determine whether strands cross over or under the others. Since the profile braid is determined by the stack sequence, we need to be able to denote throws from the back from throws from the front. We'll use α_i and ω_j to denote a throw from the back and a throw from the front, respectively. The subscript refers to the height of the throw in stack notation. Figure 6 is an example of a back throw and a front throw in a five-ball juggling pattern. Both throws in Figure 6 correspond to a 4 in the stack sequence.

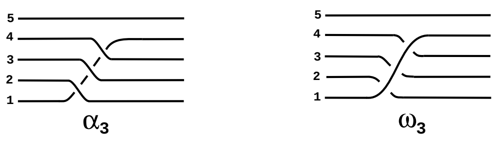


Figure 6: The two types of one-handed juggling throws.

Notice that α_i and ω_i can be expressed as

$$\alpha_i = \sigma_1\sigma_2\ldots\sigma_i$$
$$\omega_i = \sigma_1^{-1}\sigma_2^{-1}\ldots\sigma_i^{-1}.$$

We'll call the set of words generated by all jugglable braids with b balls M_n and M_n^+. Elements in M_n are those that can be expressed as words in the α_i's and elements in M_n^+ are those that can be expressed as words in the α_i's and ω_j's. M_n and M_n^+ are monoids. Neither M_n nor M_n^+ forms a group because not all elements have an inverse, but they satisfy the rest of the properties of a group.

3 Classifying Juggling Braids

It is not always possible to solve the word problem given a group with generators and relations. In many cases there does not exist an algorithm to determine if two words are equivalent that will terminate in finite time. This makes the task of classifying unbraids seem daunting. However, we shall try to make progress by looking at certain families of braids.

3.1 Unbraids

A non-trivial unbraid is a word of at least one generator that is equivalent to the unbraid. A natural question that arises about the monoids M_n and M_n^+ is whether or not they contain any non-trivial unbraids. It is not difficult to show that M_n does not contain any non-trivial unbraids. Every element except the identity in M_n has at least one pair of strands (i, j) such that $\mathrm{cr}(i, j) > 0$. And it is impossible to get any pair of strands to have a negative crossing number. In order to get a non-trivial unbraid, the sum of the crossing numbers of every pair of stands must be zero. This is impossible using just words in the α_i's.

However, this argument does not work for M_n^+. Right away we see that $\alpha_1\omega_1$ is an unbraid, and we can concatenate this to itself to get an infinite family of unbraids. In fact, these are not the only unbraids in M_n^+. One such example, the braid $\omega_1\alpha_2\alpha_2\omega_2\omega_1$ in M_3^+, is shown in Figure 7.

We wish to classify all such unbraids. We shall start by looking at patterns in M_3^+, or in other words, three ball patterns allowing front and back throws.

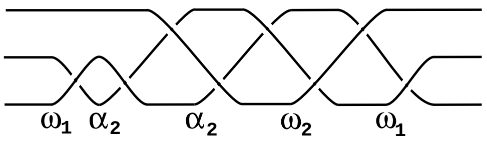
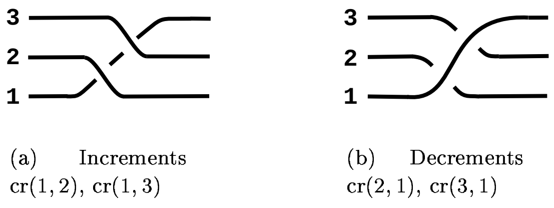


Figure 7: A non-trivial unbraided juggling pattern: $\omega_1\alpha_2\alpha_2\omega_2\omega_1$.

In the remainder of the paper, a juggling pattern will be assumed to be of this type unless otherwise stated.

A braid will be unbraided if and only if for all distinct pairs of strands i and j, $\text{cr}(i,j) = \text{cr}(j,i) = 0$. In a 3-braid, we have six different crossing numbers. Each α_i or ω_i will change exactly i crossing numbers by ± 1. Suppose the balls are numbered #1,#2,and #3. An α_i crosses under the first i strings from the below, so this increments each $\text{cr}(1,j)$ by 1 for all $j \leq i$, assuming that the bottom ball is labeled ball 1. An ω_i crosses over the first i strings, so each of these strings crosses under the bottom string from above. Thus all crossing numbers $\text{cr}(j,1)$ are decremented by 1.



(a) Increments $\text{cr}(1,2)$, $\text{cr}(1,3)$

(b) Decrements $\text{cr}(2,1)$, $\text{cr}(3,1)$

Figure 8: How α_2 and ω_2 change the crossing numbers.

The crossing numbers that get changed are dependent not only on the type of throw, but also on the current permutation of the braid. An example is given in Figure 8. If the permutation of the balls from bottom to top is 123, and the next throw is an α_2, then ball 1 crosses behind the paths of ball 2 and ball 3. This increments $\text{cr}(1,2)$ and $\text{cr}(1,3)$. However, if the permutation of the balls had been 213, then ball 2 would have crossed behind the paths of ball 1 and ball 3. A subsequent α_2 would have instead incremented $\text{cr}(2,1)$ and $\text{cr}(2,3)$. Figure 3.1 shows a table that denotes how the throws affect the crossing numbers based on the permutation.

This table shows how α_2 and ω_2 throws affect the crossing numbers of the braid given its current permutation. A "+" in an entry means that the crossing number in that column is incremented by one if the braid permutation is one of the two in that row. Likewise, the "–" means the crossing number is decremented by one. The α_1 and ω_1 throws are much simpler. Since such a

Permutations	Throw	cr$(1,2)$	cr$(2,1)$	cr$(1,3)$	cr$(3,1)$	cr$(2,3)$	cr$(3,2)$
123, 132	α_2	$+$		$+$			
213, 231	α_2		$+$			$+$	
312, 321	α_2				$+$		$+$
123, 132	ω_2		$-$		$-$		
213, 231	ω_2	$-$					$-$
312, 321	ω_2			$-$		$-$	

Figure 9: How α_2's and ω_2's affect crossing numbers.

throw simply switches the bottom two balls, only two crossing numbers can be affected. If the braid permutation is ijk, then an α_1 will increment $\mathrm{cr}(i,j)$ and an ω_1 will decrement $\mathrm{cr}(j,i)$. In both cases, the resulting permutation is jik. In conclusion, a single throw will have one of the following effects on the crossing numbers:

1. Increment any one of the six crossing numbers.
2. Decrement any one of the six crossing numbers.
3. Increment two crossing numbers, namely one of the following pairs: $\{\mathrm{cr}(1,2), \mathrm{cr}(1,3)\}$, $\{\mathrm{cr}(2,1), \mathrm{cr}(2,3)\}$, or $\{\mathrm{cr}(3,1), \mathrm{cr}(3,2)\}$.
4. Decrement two crossing numbers, namely one of the following pairs: $\{\mathrm{cr}(2,1), \mathrm{cr}(3,1)\}$, $\{\mathrm{cr}(1,2), \mathrm{cr}(3,2)\}$, or $\{\mathrm{cr}(1,3), \mathrm{cr}(2,3)\}$.

The information in the table above can be encoded in a graph called a *stack graph*. The stack graph of a b-ball juggling pattern has $b!$ vertices – one for each braid permutation. There is a directed path from a vertex v_i to v_j if and only if it is possible to get from the permutation of v_i to the permutation of v_j by throwing an α_k or ω_k where $k < b$. Algebraically, this means that there is an element s of the symmetric group S_b of the form $(k\ k{-}1\ \ldots\ 2\ 1)$ such that $s : v_i \to v_j$. The stack graph of all three ball patterns is shown in Figure 10.

Each path in the stack graph has two lables which describe how the crossing numbers can change with each throw, as described in the table in Figure 3.1. For every edge traversed, we must choose whether the throw will be an α or an ω. For example, starting from the 123 vertex, there are two ways to get to 231: either throw an α_2 or an ω_2. The α_2 is denoted by $\mathrm{cr}(1,*)+$, which means that we increment $\mathrm{cr}(1,2)$ and $\mathrm{cr}(1,3)$. The "*" is a wild-card. Likewise, the ω_2 is denoted by $\mathrm{cr}(*,1)-$, which means that $\mathrm{cr}(2,1)$ and $\mathrm{cr}(3,1)$ are decremented.

Any three-ball juggling pattern can be represented as a walk on the stack graph. Moreover, pure braids have the nice property that they must be a cycle on the stack graph. This makes the task of classifying all unbraids easier. Readers familiar with the mathematics of juggling might notice a resemblance

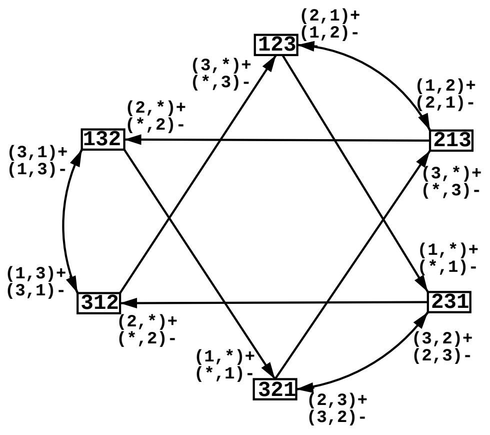


Figure 10: The stack graph for three-ball juggling patterns.

between a stack graph and the state graph, which describes when two siteswap patterns can be concatenated to form a new pattern. These two graphs have some resemblances. In both graphs, vertices represent some kind of state, and edges represent throws. Siteswap patterns correspond to closed loops on the state graph, whereas any path on the stack graph corresponds with a siteswap pattern. However, state graphs and stack graphs describe two completely different aspects of siteswap patterns. A great source for learning all about state graphs is [3].

The stack graph displays a good deal of symmetry. There are two types of edges: each vertex has one "long" edge, corresponding with an α_2 or ω_2, going into it and one going out of it. Also, each vertex has one "short" edge, corresponding with an α_1 or ω_1, going into it and one short edge leaving. Next we will present several ways to set up a system of equations whose solutions will describe all unbraids.

3.2 Setting the crossing numbers to zero.

Without loss of generality, assume that any three ball juggling pattern begins with the permutation 123. If we keep a running total of the sum of all six crossing numbers, then unbraids will be precisely those cycles such that the the crossing numbers is zero. There are six pairs of crossing numbers that can be changed with a single throw, as well as all six individual crossing numbers that can be changed independently. Thus there are twelve possible non-empty subsets of

$$\{\mathrm{cr}(1,2),\mathrm{cr}(2,1),\mathrm{cr}(1,3),\mathrm{cr}(3,1),\mathrm{cr}(2,3),\mathrm{cr}(3,2)\}$$

that can be changed by a single throw. An unbraid has the restriction that each of the crossing numbers is zero. This gives us a system of six equations on twelve variables, which we can represent by the following matrix:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Each row of the matrix represents a crossing number, and each column in the matrix represents a way to change the crossing numbers. Observe that the first six columns in 1 are the six rows in Table 3.1. Elements in the nullspace of C describe ways to traverse edges in the stack graph so that the sum of each crossing numbers is zero. However, it is important to notice that such an element might not necessarily be a closed path, which means that it physically cannot be juggled. The nullspace of C is six-dimensional, with basis $\{X_1, X_2, X_3, X_4, X_5, X_6\} =$

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}. \tag{2}$$

Pictorially, these six elements may be realized on the stack graph as in Figure 11.

Obviously, X_5 and X_6 are elements in the nullspace that are not realizable juggling patterns. However, it is important to understand that each basis element has several valid realizations on the stack graph.

Definition 3.1 *An unbraid class is a twelve-dimensional vector $\mathcal{U}$ that can be expressed as a linear combination of X_i's such that the first six entries of $\mathcal{U}$ are non-negative.*

The non-negative condition ensures that there is a realization on the stack graph. For example, the vector $-X_1$ is in the nullspace of C but is not an unbraid class because there is no physical meaning to making a negative number of throws in a juggling pattern.

We shall call a realization of an unbraid class a *loop* if it contains a cycle using all of its edges. Otherwise, it shall be called a *fragment.* The basis elements X_5 and X_6 can only be realized as fragments, whereas the other four can be realized as fragments or loops. Figure 12 shows X_1 realized three different ways. The first one is a fragment and the last two are loops.

3.3 Combinatorial questions

Unbraid classes bring up a lot of interesting combinatorial questions that are not the focus of this paper and have never been looked at closely. We shall

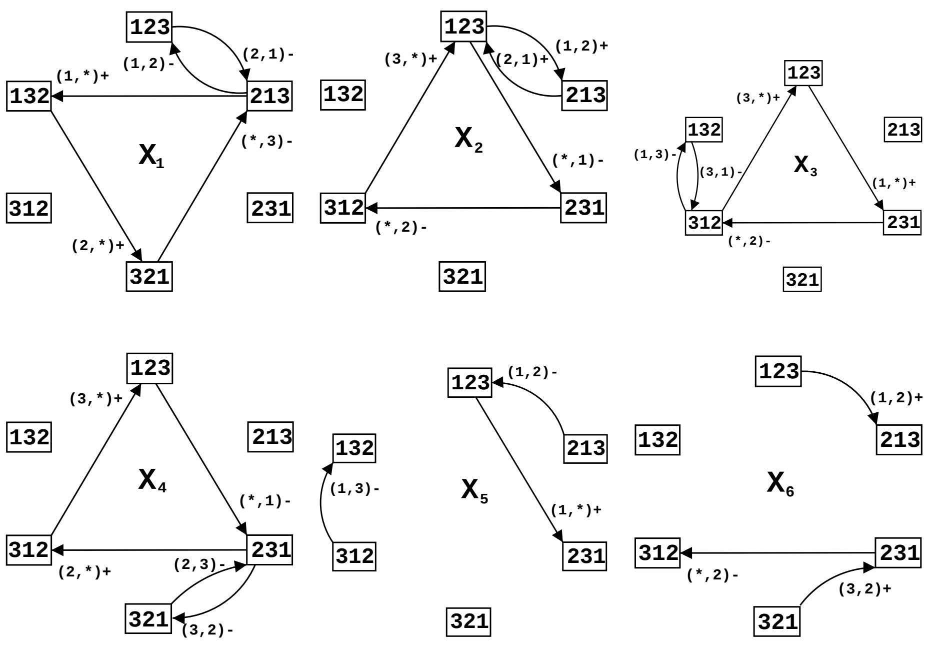
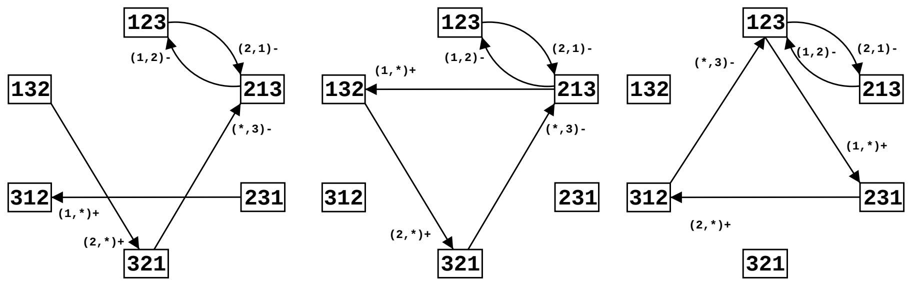


Figure 11: Realizations of the basis for all unbraids on the stack graph.



(a) Fragment (b) Loop (c) Another loop

Figure 12: Different realizations of the basis element X_1.

mention them here for sake of completeness, and to hopefully spark interest in this new topic. It is easy to see that the unbraid class X_1 has eight possible realizations, two of which are loops. Each of these loops correspond with five throws. We shall call the minimal length of a loop of an unbraid class the length of the unbraid class. Here are a few interesting unknown questions about unbraid classes:

- How many length-n unbraid classes are there?
- How many distinct loops can arise from a length-n unbraid class?
- How many ways are there to traverse a loop of a length-n unbraid class?
- How many juggling patterns can arise from a length-n unbraid class (in other words, the number of ways to traverse up to equivalence)?

To illustrate the subtle differences in what these problems are counting, consider a simple example, the length-5 unbraid classes. It is not hard to show that there are only four, namely X_1, X_2, X_3, and X_4. As mentioned above, there are two distinct loops of X_1. There are three ways to traverse, or walk, these two loops: $\omega_1\alpha_2^2\omega_2\omega_1$, $\omega_1^2\alpha_2^2\omega_2$, and $\alpha_2^2\omega_2\omega_1^2$, which are all equivalent because they differ by cyclic shift. Therefore, only one distinct juggling pattern arises from X_1. Checking the other length-5 unbraid classes, one can conclude that there are four length-5 unbraid classes, and each can give rise to at most two loops, three walks, and one juggling pattern.

3.4 Shortcomings of unbraid classes

As mentioned earlier, every realization of X_5 or X_6 is a fragment. However, this does not mean that there are no unbraid classes that contain some number of these elements. One necessary property of a juggling pattern on the stack graph ignored by the equations in (1) and unbraid classes is that the in-degree of any vertex equals its out-degree. The problem lies in the fact that for each possible pair of crossing numbers that can be incremented or decremented together, there are *two* different edges that can do this. In addition, each edge can correspond to two different variables in (1), because each edge has a positive and negative label that correspond with α and ω throws, respectively. Each of the twelve edges has two different labels, so a throw in a juggling pattern can correspond to moving on the stack graph one of twenty-four possible ways.

3.5 The Complete System of Equations

Suppose we label the edges of the stack graph as shown in Figure 13. The short double edges are actually two edges, and are labeled as such. For example, e_1 is the edge from vertex 123 to vertex 213, while e_2 is the edge from vertex 213 to vertex 123.

However, we want to be able to distinguish between α and ω throws. For each edge i, let i_+ represent traversing that edge by an α and i_- represent traversing that edge by an ω. For example, the a_+ edge is the edge from vertex 123 to vertex 231 where $\mathrm{cr}(1,2)$ and $\mathrm{cr}(1,3)$ are incremented. On the other hand, a_- is that same edge, only $\mathrm{cr}(2,1)$ and $\mathrm{cr}(3,1)$ are decremented. Recall that the twelve columns of the matrix in (2) represented the twelve ways two change the crossing numbers, and we set up six equations that represented the sum of

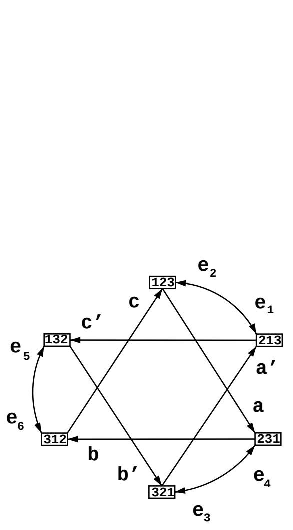


Figure 13: Labeling the edges of the stack graph.

each of the six crossing numbers vanishing. We can use the same equations as represented by the matrix C, in (1), but using these new variables. Moreover, we can eliminate a lot of solutions that do not correspond to juggling patterns by ensuring that the out-degree of each vertex equals its in-degree. This additional requirement gives us the following six equations:

$$\begin{array}{rl}
\mathbf{123}: & (e_{1+}+e_{1-})+(a_{+}+a_{-}) = (e_{2+}+e_{2-})+(c_{+}+c_{-}) \\
\mathbf{213}: & (e_{2+}+e_{2-})+(c'_{+}+c'_{-}) = (e_{1+}+e_{1-})+(a'_{+}+a'_{-}) \\
\mathbf{231}: & (e_{3+}+e_{3-})+(b_{+}+b_{-}) = (e_{4+}+e_{4-})+(a_{+}+a_{-}) \\
\mathbf{321}: & (e_{4+}+e_{4-})+(a'_{+}+a'_{-}) = (e_{3+}+e_{3-})+(b'_{+}+b'_{-}) \\
\mathbf{132}: & (e_{5+}+e_{5-})+(b_{+}+b_{-}) = (e_{6+}+e_{6-})+(c'_{+}+c'_{-}) \\
\mathbf{312}: & (e_{6+}+e_{6-})+(c_{+}+c_{-}) = (e_{5+}+e_{5-})+(b_{+}+b_{-})
\end{array} \tag{3}$$

Together, with (3) gives us a twelve equations on twenty-four variables. However, in all twelve equations, e_{1+} appears on one side of the equation if and only if e_{2-} is on the other side of (3). Likewise, e_{2+} and e_{1-} also come in pairs. In fact, each of the e_{i+} has a corresponding e_{j-}. The following six variables can be substituted

$$\begin{array}{lll}
E_1 = e_{1+} - e_{2-} & E_3 = e_{3+} - e_{4-} & E_5 = e_{5+} - e_{6-} \\
E_2 = e_{2+} - e_{1-} & E_4 = e_{4+} - e_{3-} & E_6 = e_{6+} - e_{5-}.
\end{array} \tag{4}$$

This simplification leads to a 12×18 matrix with nullspace

$$\left\{ \begin{bmatrix} 0\\1\\0\\1\\1\\0\\0\\0\\0\\0\\0\\0\\1\\1\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\1\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\1\\0\\0\\1\\0\\0\\0\\0\\0\\0\\-1\\-1\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\1\\1\\0\\1\\0\\0\\1\\0\\0\\0\\0\\-1\\0\\0\\-1\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\1\\0\\1\\0\\1\\1\\0\\1\\0\\1\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\1\\1\\0\\0\\1\\1\\0\\1\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\1\\1\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\-1\\-1\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\1\\1\\0\\0\\1\\1\\0\\0\\0\\0\\0\\-1\\0\\0\\-1\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\1\\1\\1\\0\\1\\1\\0\\1\\0\\0\\0\\-1\\0\\0\\0\\-1\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\0\\1\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\-1\\-1 \end{bmatrix} \right\} \tag{5}$$

The nullspace has a basis with the nice property that the last six non-zero entries of the vectors can be either positive or negative and still be realized on the stack graph. Also, right away we see that there are unbraids that use the basis elements X_5 and X_6. Notice that the sixth and seventh columns in (5) correspond to length-four unbraids, as shown in Figure 14. The braid shown in Figure 14 is an example, it is a walk of the unbraid class $X_1 + X_2 - X_4 - X_6$.

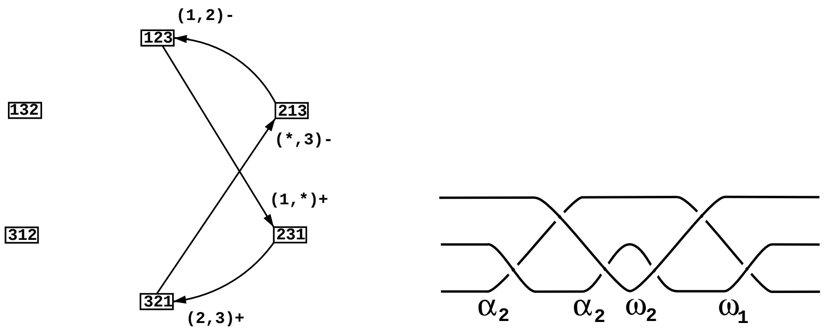


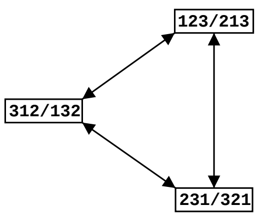
Figure 14: A length-four unbraid.

The basis in (5) has several advantages over the basis in (2). First of all, each basis vector in (5) is a cycle on the stack graph. We no longer have to worry about fragments because each entry in the vectors in (5) refers to a specific edge on the stack graph. This means that any linear combination of basis vectors can be juggled if the first twelve entries are non-negative. However, even though there is only one realization for each linear combination of basis elements on the stack graph, there may be several different possible orders to traverse the edges. In the original basis, two of the vectors were fragments and could not be concatenated with other vectors at will. However, in both bases, there are unbraids that cannot be represented as positive linear combinations of the basis vectors. It would be nice to find a smallest set of vectors, if such a finite set exists, in both (2) and (5) such that any unbraid can be represented as a positive linear combination of basis vectors.

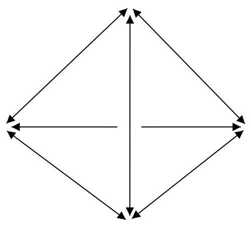
Another natural step is to examine not just unbraids, but also look into when two three-ball juggling patterns yield the same braid, and when two paths on the stack graph correspond with the same braid. Eventually, it would be interesting to look at stack graphs of patterns of more than three-balls and see if similar results hold. Proof techniques must be generalized, because the size of the stack graphs grow large very quickly. The number of vertices in the graph of a b-ball pattern is $b!$.

4 Adding More Balls

Thus far, we have not examined the stack graphs of patterns with more than three balls because the size of the graph grows very quickly. A good way to understand a larger stack graph is to collapse it into a smaller graph. For any stack graph, identify two vertices if they have the same top ball in their permutation, and remove all singleton edges. This new graph is called the *condensed graph.*



(a) the three-ball condensed graph


(b) the four-ball condensed graph

The labels of the vertices are omitted from the four-ball condensed graph for clarity in Figure 4. It is easy to see why the n-ball condensed graph is an n-simplex. The condensed graph is a good way to visualize what the stack graphs look like with more balls. The idea is similar to how to construct a cube from a square, a hypercube from a cube, and so forth. Observe how the three-ball stack graph is constructed from the two-ball stack graph, which is just two vertices and two edges. To construct the four-ball stack graph, put one copy of the three-ball stack graph at each vertex of a 3-simplex, and add edges to represent going between these stack graphs. Each of the twenty-four vertices has one directed edge *to* a vertex in each of the other three-ball stack graphs, and one directed edge *from* a vertex in each of the three-ball stack graphs. Notice how all of these forty-eight edges correspond with the α_3 and ω_3 throws.

5 Further Research

The stack graphs presented in this paper represent one-handed juggling, where throws can be made from either side. Idealy, it would be nice to model typical two-handed juggling, where there are two throwing locations and two catching locations. Typically, a juggler makes the catches torwards the outside of a pattern and makes the throws nearer the middle of the pattern. Modeling this with the stack notation would require two stacks that interacted. Because these ideas are so new, this has not been examined much. Indeed, the problem of classifying braids of juggling patterns with one hand has proven to be a difficult task. One goal of this paper is to share a new method of describing juggling patterns and pose a number of interesting questsion that arise, with the hopes that it will spark the interest of other mathematicians. There are many different ways to generalize one-handed juggling. A popular type of juggling is bounce-juggling. More braids can arise from juggling patterns if throws can be tossed or bounced.

There are just a few of the ideas about this topic that may be worth exploring in the future.

References

[1] J. Buhler, D. Eisenbud, R. Graham, C. Wright, Juggling Drops and Descents, *American Mathematical Monthly*, Volume 101, Issue 6 (Jun.-Jul., 1994), 507-519.

[2] K. Murasugi, B. I. Kurpita, *A Study of Braids*, Kluwer Academic Publishers, 1999.

[3] B. Polster, *The Mathematics of Juggling*, Springer-Verlag, 2003

[4] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://www.research.att.com/njas/sequences/