Senior Thesis in Mathematics

An Investigation of Human Knots and Links

Author:
Daniel Rosen

Advisor:
Dr. Vin de Silva

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 28, 2020

Contents

1 Introduction **3**

2 Solving a human knot instance **6**
2.1 Constructing the star $\bigstar(n, k)$ 6
2.2 Turning $\bigstar(n, 2)$ into a knot 6
2.3 Which members of $\bigstar(n, 2)$ are the unknot? 8
2.4 Constructing a Knot Invariant 11
2.5 Proving the Knot Invariant is invariant 12
2.6 Applying this knot invariant to our knots 16
2.7 Which members of $\bigstar(n, 2)$ are the unlink? 19
2.8 Constructing a Link Invariant 20

3 Evaluating $\bigstar(7, 3)$ **25**
3.1 Enumerating the Knot 26
3.2 Reduction to $\bigstar(5, 2)$ 26
3.2.1 An Example Reduction 26
3.2.2 Detailing the Reduction Process 29
3.3 Evaluating Result 35

Appendices **37**

A Implementing the Reduction and Invariant in Sage **38**
A.1 Coding the reduction procedure 38
A.2 Constructing Knot Incidence Matrix 39
A.3 Assigning crossings 40
A.4 Retrieving every crossing assignment 41
A.5 Looping through every crossing assignment 42

Abstract

The motivation for this study of knots and links comes from the human knot game. We construct knots and links based on circumscribed stars, assigning crossings as if edges were arms stacked above and below each other, in an imitation of the knots arising from the human knot game. To solve these knots, we make use of simple coloring invariants to differentiate knots from the unknot, as well as develop strategies to simplify the analysis of more complex knots with up to 14 crossings. Finally, we computationally execute those strategies and evaluate invariants on a large class of knots with significant complexity, making them difficult to analyze manually.

Acknowledgements

This thesis would not exist without the help of my fantastic advisor, Professor Vin de Silva. Throughout the entire process, he encouraged me to keep moving forward, and I am so thankful for his oversight and guidance. These last few months have been so different from what I anticipated them being, and I was immensely grateful for the normalcy of meeting (even if over Zoom) every Tuesday morning. I also owe thanks to the many other professors who have instructed me at Pomona College, among them Professor Shahriari, Professor Flapan, and Professor Garcia.

I'd like to thank my parents, Marc and Hadassah. You really did know what you were doing when you gave me math problems for dessert all those years ago! Your continued support and love allows me to be the best I can be.

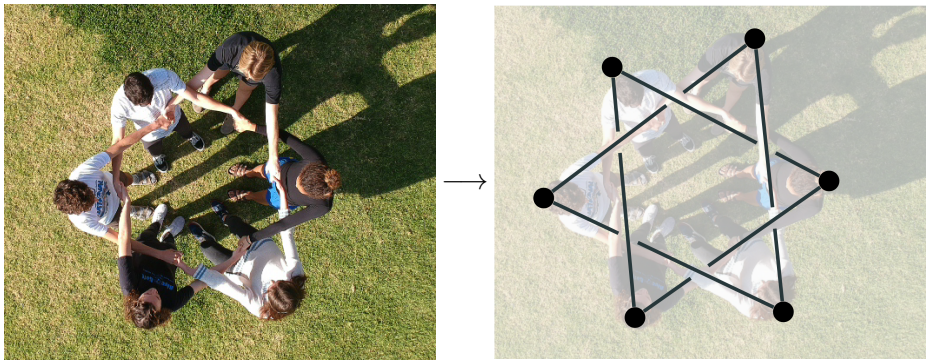
A huge thank you to my friends and roommates who've supported me throughout this process: Losing such a nurturing environment is the quickest way to realize how much it helped you in the first place. I can't wait to see you all again.

And to my PPXC teammates and coaches: SKEW! Thank you for being a beacon of light throughout my 4 years at Pomona College. My college experience wouldn't have been half of what it was if not for the folks that make up our team.

Chapter 1

Introduction

The human knot game is a common team-building activity played by small groups. The game is played as such: people stand in a circle, and link arms with two others across the circle. Once everyone has linked arms, the group attempts to disentangle the knot that they have created. A question arises: Given n people, how frequently will the knot resulting from the human knot game be isotopic to the unknot; that is to say, what is the probability that the group can disentangle the knot they've created?



This question requires us to ascertain when knots can be disentangled. How can we go about doing such a thing? It will help to define what a knot is, in a mathematical sense.

Definition 1.0.1. *A knot is a continuous mapping of the circle, S^1, into $\mathbb{R}^3$.*

The simplest knot is the unknot, or a knot with 0 crossings. This knot is just the inclusion mapping of S^1 in $\mathbb{R}^3$. In this thesis, we will also be

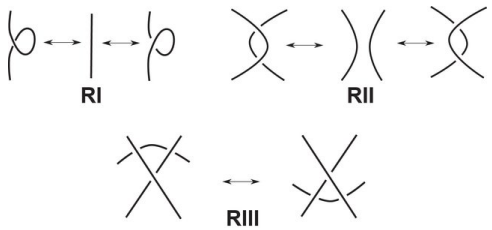


Figure 1.0.1: The 3 Reidemeister moves [2]

looking at links, which are collections of such functions whose components may interlock and interweave. The function defining a knot or those defining a link may also be represented as an oriented diagram. In this paper, we will treat knots as unoriented.

When comparing two knot diagrams, it is useful to know if those diagrams are representative of the same knot. If the first diagram can be manipulated into the second diagram without "breaking" the continuity of the mapping, then the diagrams are representative of the same knot [5].

It turns out that there are only 3 classes of manipulations on knot diagrams necessary to transform one diagram into another. These are called the Reidemeister moves. All permutations of a knot diagram obtained by applying Reidemeister moves are equivalent. The three Reidemeister moves can be viewed in Figure 1.0.1.

One useful tool that has arisen to distinguish knots is a knot invariant. A knot invariant is a mapping from K, the space of knot diagrams, to any type of value, which has the same value on diagrams of the same knot. That is, for any $a, b \in K$ which are representative of the same knot, a function f is a knot invariant if f(a) = f(b). To verify that a function f is a knot invariant, one need only check that its value is unchanged when applying the Reidemeister moves to an arbitrary knot diagram.

These invariants can be quite useful. If an invariant provides a different value for two knot diagrams, one can be certain that those knots are not reconcilable through Reidemeister moves; that is, the knots are fundamentally different objects.

Knot invariants can range widely in complexity. On the more complex side, there are knot polynomials. These knot invariants take a knot diagram and output a polynomial value. The first such polynomial was discovered by J. W. Alexander in 1928. Known as the Alexander polynomial, it is calculated algorithmically based on the regions and crossings of a knot diagram [1]. Another such polynomial is the Jones polynomial, whose discovery

in 1984 proved hugely influential for the field of knot theory, resulting in Jones winning a Fields medal [3]. However, knot invariants can be simpler. They need only remain constant regardless of representation of the knot.

This paper will examine a particular instances of small human knots and determine the conditions under which these knots are equivalent to the unknot. In doing so, we will construct and apply simple knot and link invariants in order to distinguish our knots, with the goal of making generalizations on these knots based on the crossing arrangement provided by the human arm crossings. In particular, this paper examines knots arising from the placement of vertices around a circle, with connections between adjacent or second-degree adjacent vertices.

Chapter 2

Solving a human knot instance

This chapter will introduce a particular version of the human knot game, where participants have a pre-defined behavior as to whom they grab. Then, we introduce uncertainty by allowing the arms to stack in arbitrary orderings. This reduces some of the difficulty of the problem by giving it more defined boundaries while still allowing for significant complexity. We'll start by introducing this knot format more formally, and continue by fully solving these variants of knots.

2.1 Constructing the star $\bigstar(n, k)$

In this section we consider the knot or link resulting from n people standing on the circumference of a circle. We will look at cases where the ith person in the circle joins hands with the $i + 2$th and $i - 2$th people on the circle. This results in an n-pointed star circumscribed in the circle.

In general, we call the figure resulting from n people placed around a circle with the ith person joining hands with the $i+k$th and $i-k$th persons $\bigstar(n, k)$. So, the arrangement we described above falls can also be written a s$\bigstar(n, 2)$. Some of these stars are depicted in Figure 2.1.1.

2.2 Turning $\bigstar(n, 2)$ into a knot

To turn $\bigstar(n, 2)$ this into a knot, we must assign crossings to the intersections of the star created. Any of these arrangements would be possible with a human knot, since the humans would essentially be choosing the vertical

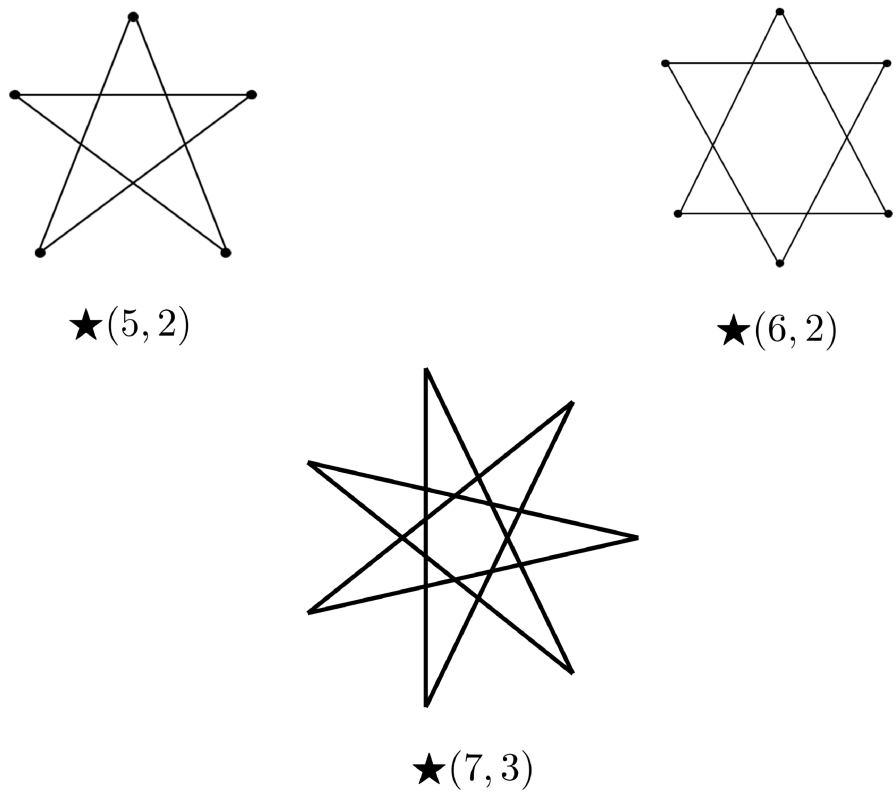


Figure 2.1.1: Examples of $\bigstar(n, k)$

order of partners to link arms with at random. Figures 2.2.1 and 2.2.2 depict these arrangements with crossings assigned.

In order to do this in a more systematic way, we introduce notation with which we can discuss each crossing relative to the internal polygon of the star. If the left-most arc when facing away from the inner polygon goes *under* the right-most arc, we call it an A crossing. If the left-most arc when facing away from the inner polygon goes *over* the right-most arc, we call it a B crossing. These are illustrated in the figures 2.2.3 and 2.2.4.

Now that we have a way to label each crossing, we can refer to specific knot diagrams as an enumeration of their crossings. Starting from the top-most vertex, we go through the crossings counter-clockwise to construct a string for a given knot diagram.

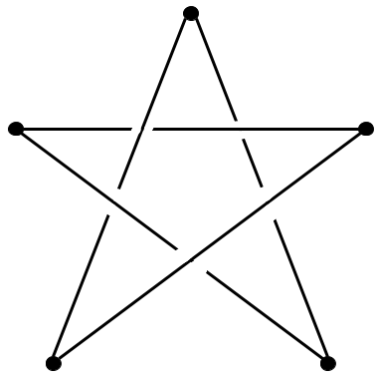


Figure 2.2.1: A knot, $n = 5$

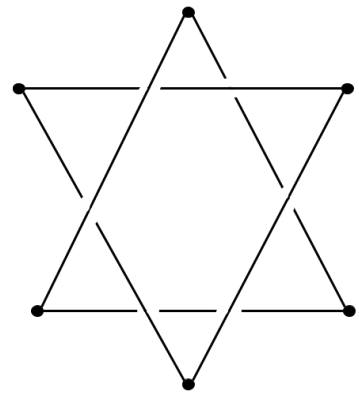


Figure 2.2.2: A link, $n = 6$

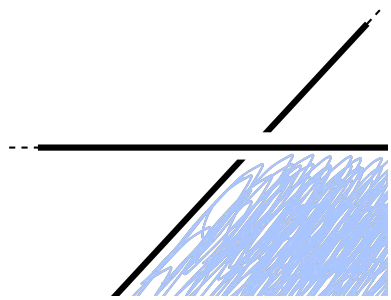


Figure 2.2.3: Crossing type A

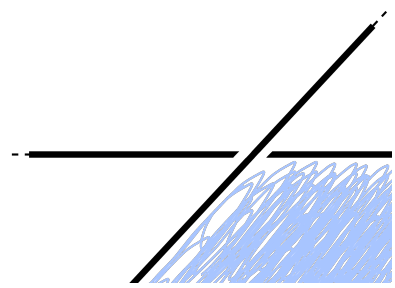


Figure 2.2.4: Crossing type B

2.3 Which members of $\bigstar(n, 2)$ are the unknot?

With this language to describe knots, we are able to express a theorem to determine which human knots of this arrangement are knotted, and which are equivalent to the unknot.

Theorem 2.3.1. *Consider a knot of the general form $\bigstar(n, 2)$, with crossings classified as A or B according to their orientation with respect to the inner polygon of the knot. Let $|A|$ and $|B|$ denote the number of A crossings and B crossings. If the number of vertices is odd, then the human knot is the unknot if and only if $|A| - |B| = \pm 1$.*

We will need to develop additional machinery to rigorously prove this claim.

We begin with a lemma to reduce the number of crossing assignments

we need to look at, from 2^n assignments to a much smaller number.

Lemma 2.3.2. The Crossing Swapping Lemma:
Given a string X, if X contains a substring AB, then this substring can be replaced by the substring BA.

Proof. We manipulate the knot diagram using the Reidemeister moves. If an A crossing is adjacent to a B crossing, we may pull the outer vertex under the arc of the inner polygon as illustrated in Figure 2.3.1, and then push it back outwards over the arc of the inner polygon.

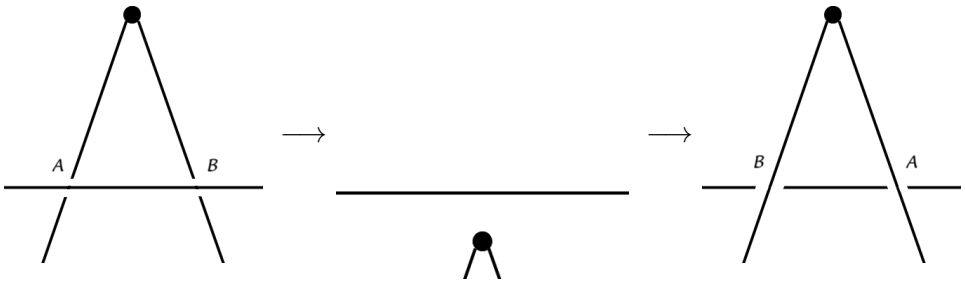


Figure 2.3.1: Crossing swapping lemma

□

This is a simple argument but it leads to a powerful result allowing us to reduce the total number of representations of these cases of knots.

Corollary 2.3.3. *If two strings have the same number of A's and B's, then they describe the same knot.*

Proof. Use the Crossing Swapping Lemma (Lemma 2.3.2) to rearrange As and Bs. If the number of As and Bs are the same, then the string can be reordered into the same ordering to obtain the same knot diagram, and thus the same knot. □

Example 2.3.4.

$$
\begin{array}{ccccccc}
A & A & A & A & B & B & B \\
A & A & A & B & A & B & B \\
A & A & B & A & A & B & B \\
A & B & A & A & A & B & B \\
A & B & A & A & B & A & B \\
A & B & A & B & A & A & B \\
A & B & A & B & A & B & A
\end{array}
$$

Since an A crossing looks like a B crossing when flipped, we may say without loss of generality that there is always a greater number of As in a string than Bs. If there were to be more Bs in a string, we simply flip the knot diagram and construct a string with more As. So in sum, the number of distinct knots we need to analyze is

$$|W_n| = \begin{cases} \frac{n}{2} + 1 & n \text{ even} \\ \frac{n-1}{2} + 1 & n \text{ odd} \end{cases}.$$

Definition 2.3.5. : Knot normal form: *Using the Crossing Swapping Lemma, we can create a normal form of strings. For a string x, we denote the number of As and Bs with $|A|$ and $|B|$, respectively. Then we can rearrange the string x into the following normal form:*

$$\overbrace{ABAB\ldots AB}^{|B|}\overbrace{AA\ldots A}^{|A|-|B|}.$$

Let $k = |A| - |B|$. Then the normal form corresponds to the diagram in Figure 2.3.2:

With the normal form in hand, we see that each of the AB substrings can be untangled in the knot diagram through a Reidemeister move, as illustrated below:

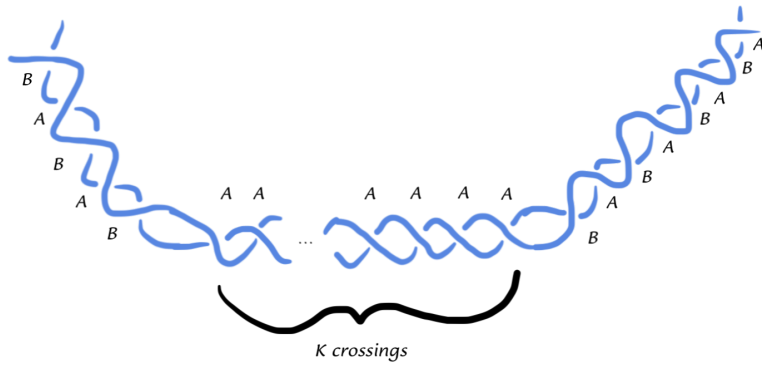
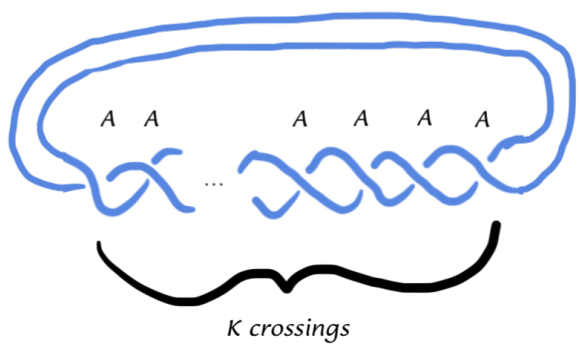


Figure 2.3.2: Knot created by normal form



When $|A| - |B| = 1$, we simply untwist the final remaining crossing to obtain the unknot. Thus, we have properly shown that knots with $|A|-|B| = 1$ are the unknot.

By spreading out these crossings throughout the outer section of the upper arc of the topmost "loop", we obtain a knot which appears to be the knot K_1 in many knot lookup tables [4]. However, this not sufficiently rigorous to prove that these knots are not the unknot. To do so we must show that *no* combination of Reidemeister moves can result in the unknotting of these diagrams. Here, we make use of an elementary knot invariant to prove that these knots are indeed distinct from the unknot.

2.4 Constructing a Knot Invariant

We'd like a knot invariant which distinguishes knots of type $\bigstar(n, 2)$. It may distinguish other kinds of knots as well, but minimally we'd like it to

distinguish knots which have differing numbers of As and Bs. To do this, we introduce the concept of coloring.

Definition 2.4.1. *An* **arc** *on an oriented knot diagram is a segment of a knot diagram between two under crossings.*

A knot diagram with n crossings will have n arcs.

Definition 2.4.2. *Let S be a set. An* **S-coloring** *of an oriented knot diagram is a mapping from the set of arcs on a knot diagram to S.*

Definition 2.4.3. *Let G be an abelian group. A* **G-coloring** *of an oriented knot diagram is a mapping from the set of arcs on a knot diagram to G.*

Definition 2.4.4. *Let F be a field. An* **F-coloring** *of an oriented knot diagram is a mapping from the set of arcs on a knot diagram to F*

For our purposes, it will be useful to use a field, since it is simple to row reduce matrices generated from this invariant within the field. So, we will simply refer to an F-coloring as a coloring.

To define our invariant, we introduce a notion of legal and illegal colorings.

Definition 2.4.5. *A* **legal coloring** *of a knot diagram is one which satisfies the following condition:*

(1) For each pair of arcs a_1, a_2 traversed by a_3 in a crossing, the assignments must satisfy the equation:

$$2a_3 = a_1 + a_2. \tag{2.4.1}$$

2.5 Proving the Knot Invariant is invariant

In this section, we will establish that the invariant arising from Equation 2.4.1 indeed does not vary under the three Reidemeister moves, which can be found in Figure 1.0.1.

Theorem 2.5.1. *Let K be a knot. Then, the number of legal F-colorings of K is invariant under each of the three Reidemeister moves.*

Proof. Let F be the field from which we assign values to the arcs in K.

(1) The first Reidemeister move consists of introducing a loop in an arc, as depicted in Figure 2.5.1. We'd like to show that there is a bijection between the colorings of the knot before the move and after the move.

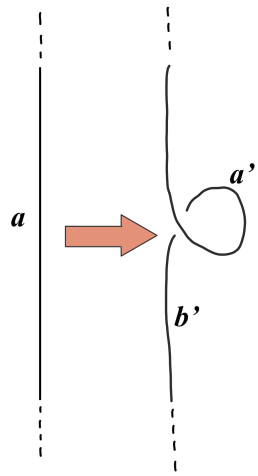


Figure 2.5.1: Labeling the first Reidemeister move.

Let C be the set of colorings of the knot before the Reidemeister move and C' be the set of colorings after the Reidemeister move. We'd like to find a function $f : C \to C'$ that is a bijection, that is, it is injective and surjective. We must define f. For a coloring c in C, let $f(c)$ be the coloring obtained by coloring all of the unpictured arcs on the right hand side with the same value as in c. Then, color a' and b' with the value a, such that $a' = b' = a$.

First we will show that f is surjective. Suppose we have a coloring c' of the knot after the Reidemeister moves. For this coloring to be legal, $2a' = a' + b'$, so $a' = b'$. We'll build a coloring c in C such that $f(c) = c'$. First, let all the unpictured arcs on the left hand side of Figure 2.5.1 take the same value as their counterparts on the right-hand side. Since these arcs and crossings have not changed, those assignments will be valid on the left hand side as they were on the right hand side. Then, let $a = a'$. Since $a = a'$, any unpictured crossing on the top of the figure will satisfy Equation 2.4.1. And, since $a = b'$, any unpictured crossing on the bottom of the figure will satisfy Equation 2.4.1. Therefore, c is a legal coloring. Since c is a legal coloring, f is surjective.

Now we must show that f is injective. Let $x, y \in C$. Let x_a and y_a be the value of arc a taken in the respective colorings. Suppose $f(x) = f(y)$. Then, all of the individual arcs on the right hand side agree in color. We know that f preserves the color of all of the unpictured arcs. We also know that $a' = x_a$ and $a' = y_a$. Thus, $x_a = y_a$, so the value of arc a is the same

in both x and y, so x and y are really the same coloring!

Therefore f is bijective, and $|C| = |C'|$.

(2) The second Reidemeister move consists of creating two new crossings by crossing on arc over another and back. We will follow the same strategy as above, constructing a bijection between the set of colorings before the move, C, and the set of colorings after the move, C'. According to the labels in Figure 2.5.2, we define a function $f : C \to C'$ which leaves all colors constant, with the exception of setting $a' = a$, $b' = b$, $d' = b$, and $c' = 2a - b$. This is indeed a legal coloring since $c' = 2a - b \Rightarrow 2a' = b' + c'$ and $2a' = c' + d'$, since $b = b' = d'$.

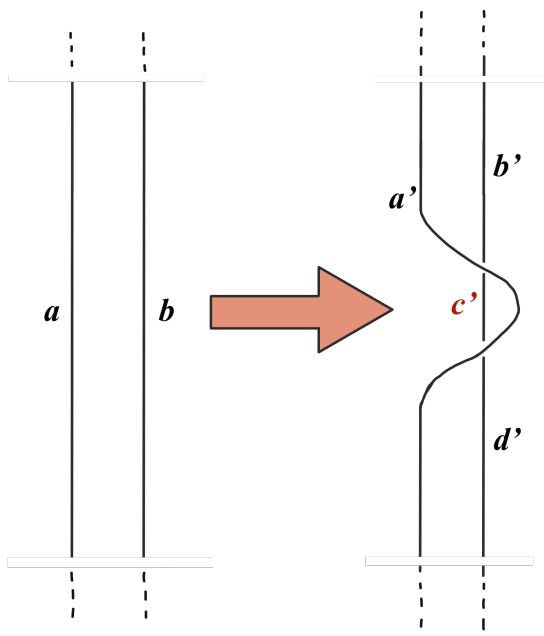


Figure 2.5.2: Labeling the second Reidemeister move.

The argument for surjectivity is the follows from the argument in (1).

To prove injectivity, see that in any coloring in C', the value c' is uniquely determined by a and b, so f is indeed injective.

Since we've constructed a bijection, $|C| = |C'|$ and the invariant is not affected by the second Reidemeister move.

(3) The third Reidemeister move is depicted in Figure 2.5.3. We aim to show that there exists a bijection. In this case, the difficult part of the proof

is actually defining the bijection. The proof that it is indeed a bijection is similar to the previous cases.

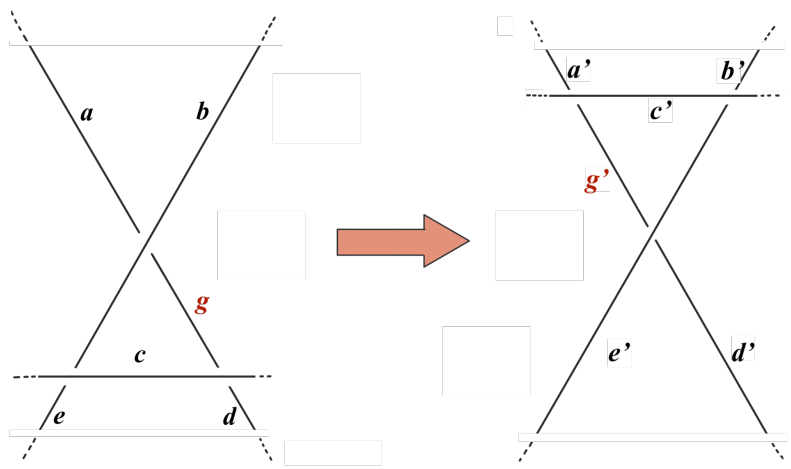


Figure 2.5.3: Labeling the third Reidemeister move.

Let C be the set of colorings on the left hand side and C' be the set of colorings on the right hand side. We define $f : C \to C'$ as coloring $a' = a$, $b' = b$, $c' = c$, $d' = d$, $e' = e$, and $g' = 2c - a$.

Starting with a legal coloring on the left hand side, we aim to show that there is a corresponding legal coloring of the right hand side.

From the left hand side, we see that

$$\begin{aligned} a + g &= 2b \\ b + e &= 2c \\ d + g &= 2c. \end{aligned}$$

On the right hand side, we must show that we get a legal coloring. To do that, we must satisfy the following equations:

$$\begin{aligned} a + g' &= 2c \\ b + e &= 2c \\ d + g' &= 2e. \end{aligned}$$

We immediately see that the middle equation on the right hand side is satisfied by the middle equation from the left hand side. The first equation

on the right hand side is satisfied as well, by the value of g'. We must work to show that the bottom equation is satisfied by our value of g', so, we want to show that $2e - d - g' = 0$ in our field F.

$$
\begin{aligned}
2e - d - 2c + a = & \\
&= 2(2c - b) - d - 2c + a && \text{substituting } e = 2c - b \\
&= 4c - 2b - d - 2c + a \\
&= 2c - 2b - d + a \\
&= 2c - (a + g) - d + a && \text{substituting } 2b = a + g \\
&= 2c - g - d \\
&= 0 && \text{By the third equation from the left hand side.}
\end{aligned}
$$

Thus, the function f produces a legal coloring on the right hand side. By previous arguments, this function is also a bijection, so $|C| = |C'|$.

Since the number of colorings of a knot does not change under the three Reidemeister moves, it is indeed a knot invariant. □

Thus we know that the number of valid colorings is invariant to the knot diagram – that is, regardless of the representation of a knot, we will always have the same number of valid colorings.

2.6 Applying this knot invariant to our knots

Assume that our field F is cyclic, so of the form $\mathbb{Z}/p\mathbb{Z}$. We'll add in labels to the above knot diagram to show that assigning arcs values from our group G results in the following system of equations:

$$
\begin{aligned}
2a_1 &= a_k + a_2 \\
2a_2 &= a_1 + a_3 \\
2a_3 &= a_2 + a_4 \\
&\vdots \\
2a_{k-1} &= a_{k-2} + a_k \\
2a_k &= a_{k-1} + a_k
\end{aligned}
$$

By adding up all of the equations on the left and right sides, respectively, we receive a tautology, $2a_1 + 2a_2 + \cdots + 2a_k = 2a_1 + 2a_2 + \cdots + 2a_k$,

which implies that the system is overdetermined. Thus we can eliminate an equation from from the system. We choose the last, giving us a new system:

$$\begin{aligned} 2a_1 &= a_k + a_2 \\ 2a_2 &= a_1 + a_3 \\ 2a_3 &= a_2 + a_4 \\ &\vdots \\ 2a_{k-1} &= a_{k-2} + a_k \end{aligned}$$

Solving the final equation for a_k, we get $a_k = 2a_{k-1} - a_{k-2}$. By inserting this value into the remaining values of a_k, we can eliminate this equation from the system. After this first step, we have the system

$$\begin{aligned} 2a_1 &= a_2 + 2a_{k-1} - a_{k-2} \\ 2a_2 &= a_1 + a_3 \\ 2a_3 &= a_2 + a_4 \\ &\vdots \\ 2a_{k-2} &= a_{k-3} + a_{k-1}. \end{aligned}$$

Theorem 2.6.1. *This system reduces to* $k(a_1 - a_2) \equiv 0 \mod p$.

Proof. We will prove this claim inductively. First, we make the claim that after h steps of eliminating equations, the top equation is

$$2a_1 = a_2 - ha_{k-h-1} + (h+1)a_{k-h}.$$

The base case was proven above, and the inductive case follows as such: Assume the inductive hypothesis, that

$$2a_1 = a_2 - ha_{k-h-1} + (h+1)a_{k-h}.$$

The system of equations at this point has eliminated the equations with $2a_i$ on the left hand side for $i \geq k - h$, the $h+1$th step of induction. So we may now eliminate the equation

$$2a_{k-h-1} = a_{k-h-2} + a_{k-h}$$

by solving for $a_{k-h} = 2a_{k-h-1} - a_{k-h-2}$. Plugging this in to the first equation, we yield

$$\begin{aligned} 2a_1 &= a_2 - ha_{k-h-1} + (h+1)(2a_{k-h-1} - a_{k-h-2}) \\ &= a_2 - ha_{k-h-1} + (2h+2)a_{k-h-1} - (h+1)a_{k-h-2} \\ &= a_2 - (h+1)a_{k-(h+2)} + (h+2)a_{k-(h+1)}. \end{aligned}$$

Thus the inductive claim is true.

Now, after $k-3$ steps, we get the following equation as the final result of our system, determining the rest of the system:

$$2a_1 = a_2 - (k-2)a_1 + (k-1)a_{k-2}.$$

Simplifying this and applying the modular arithmetic of the group we are operating within gives us

$$k(a_1 - a_2) \equiv 0 \mod p. \tag{2.6.1}$$

□

We can backtrack to find solutions for each of the k variables in the system in terms of a_1 and a_2. Since this equation determines the rest of the variables in the system, the number of solutions satisfying this equation is the same as the number of solutions to the overall system, which we recall determines the number of colorings of the knot we are concerned with.

We are now ready to prove Theorem 2.3.1. Recall the statement of the theorem:

Consider a knot K of the general form $\bigstar(n, 2)$, with crossings classified as A or B according to their orientation with respect to the inner polygon of the knot. Let $|A|$ and $|B|$ denote the number of A crossings and B crossings. If the number of vertices is odd, then the human knot is the unknot if and only if $|A| - |B| = \pm 1$.

Proof. Let $k = |A| - |B|$. We have already shown one direction, that if $k = 1$, the human knot K is the unknot. Now, with the invariant, we can prove the contrapositive of the converse, equivalent to the reverse direction of the if and only if.

We'd like to show that if $k > 1$, then K is knotted. To do this, we need to show that given k, we can find a field F which provides a different number of colorings for the unknot and the knot of interest.

Let k be given. Then, choose $p > 2$ such that p divides k. Choose the field $\mathbb{Z}/p\mathbb{Z}$. Then, see that the unknot has exactly p colorings: the single arc of the unknot can be colored by any member of the field, and it will be a legal coloring. However, since $k \equiv 0 \mod p$, no matter the values of a_1 and a_2 from Equation 2.6.1, we get a legal coloring. Therefore, there are p^2 colorings, since a_1 and a_2 can each take on any value from $\mathbb{Z}/p\mathbb{Z}$.

Since we can find a field for which the number of colorings differs between the K and the unknot, and the number of colorings is invariant under the Reidemeister moves, we have shown that if $k > 1$, K (the human knot) is knotted. $\square$

Corollary 2.6.2. *The probability that the human knot arrangement is unknotted is*

$$\frac{2\binom{n}{\lfloor n/2 \rfloor}}{2^n}.$$

2.7 Which members of ★$(n, 2)$ are the unlink?

An argument using the same normal form for links results in a likewise compelling result; however, we must use a different invariant to prove that the links formed are not the unlink.

Once the link is in normal form, we draw the knot diagram in Figure 2.7.1 to analyze our link:

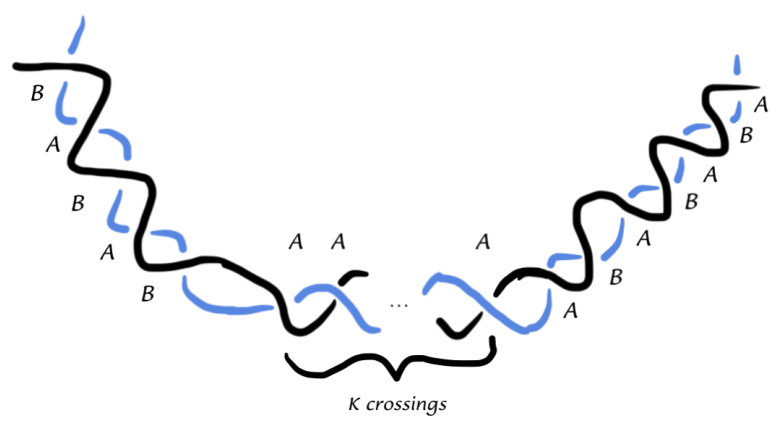


Figure 2.7.1: Link created by normal form.

By untangling the AB crossings on the wings of the diagram through a Reidemeister move, we obtain the following diagram:

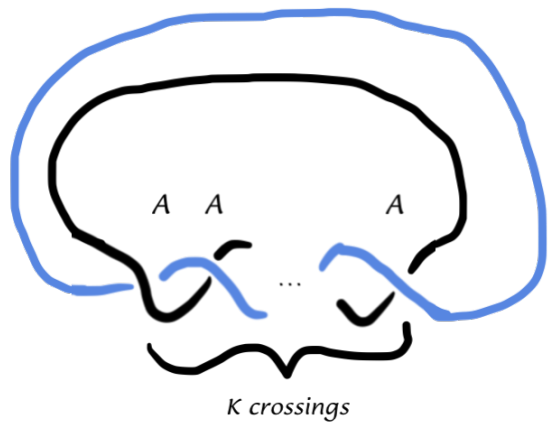


Figure 2.7.2: Link changed by Reidemeister moves.

And by pulling the lower loop below the upper loop and around to the bottom of the diagram, we obtain a link which visually appears to be inseparable.

To prove that these two links are distinct from the unlink, we make use of a link invariant.

2.8 Constructing a Link Invariant

We'd like an invariant to not change under the Reidemeister moves, and also to be able to differentiate links frequently.

In these cases, there are two disjoint loops, so instead of just A and B crossings, we must look at four types of crossings: c_1, c_2, c_3, and c_4 crossings. A depiction of these crossings can be seen in Figures 2.8.1, 2.8.4. These crossings depend on the *orientations* of the arcs which compose them, so we will invoke an orientation on the arcs of the knot diagram for the links created.

For a crossing type c_i, let $|c_i|$ denote the number of crossings of that type in a diagram. We construct a link invariant by assigning a value N_{c_i} to each crossing type. For our purposes, we assign values as such:

Then we construct our knot invariant as the following sum:

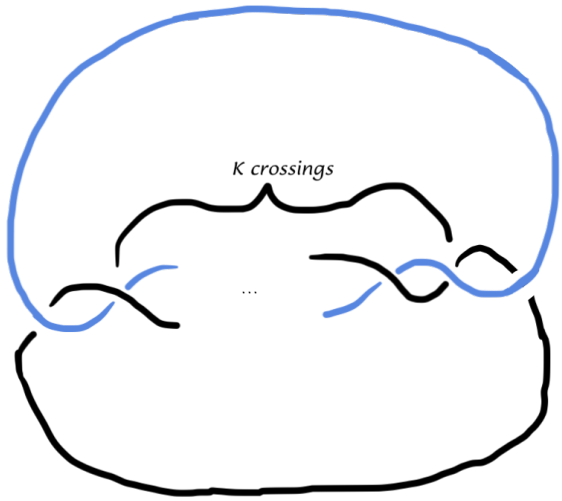


Figure 2.7.3: Another Reidemeister move to change the link.

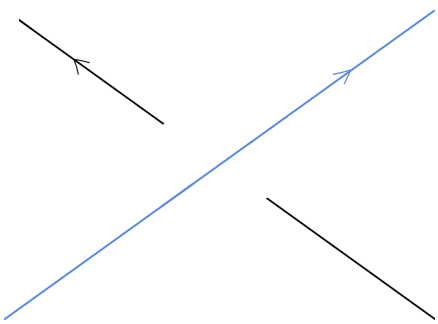


Figure 2.8.1: c_1 crossing

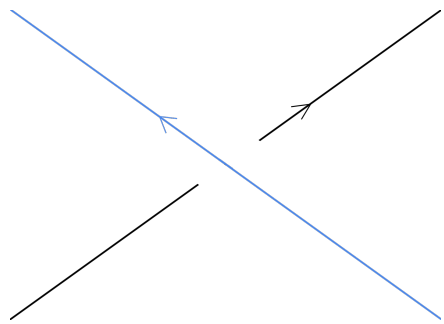


Figure 2.8.2: c_2 crossing

$$N_{c_1} * |c_1| + N_{c_2} * |c_2| + N_{c_3} * |c_3| + N_{c_4} * |c_4|. \tag{2.8.1}$$

Theorem 2.8.1. *For the values in Table 2.8.1, the expression*

$$N_{c_1} * |c_1| + N_{c_2} * |c_2| + N_{c_3} * |c_3| + N_{c_4} * |c_4|$$

is a link invariant, that is, it does not change under the Reidemeister moves.

Proof. (1) This value is trivially constant under the first Reidemeister move since there are no interactions between the knots composing the link, and the crossing types we are interested are those in which multiple knots are involved.

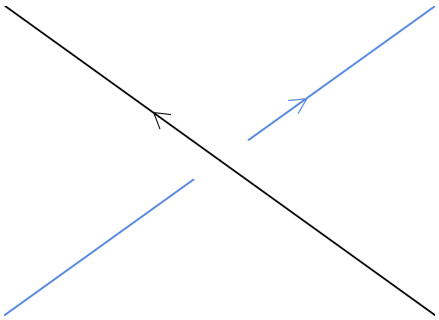


Figure 2.8.3: c_3 crossing

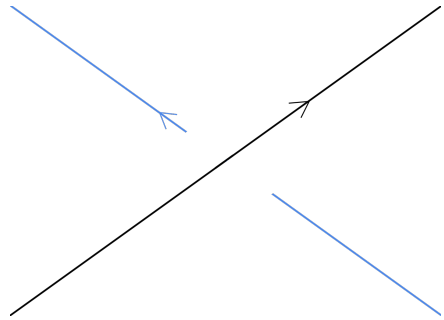


Figure 2.8.4: c_4 crossing

Crossing Type	N_{c_i}
c_1	1
c_2	-1
c_3	0
c_4	0

Table 2.8.1: Values of N_{c_i} for a candidate link invariant.

(2) Depending on orientation, the second Reidemeister move either introduces a c_1 crossing and a c_2 crossing or a c_3 crossing and a c_4 crossing. In the first case, we look at the difference between the invariant of the second link and the first link. Since we iterate $|c_1|$ and $|c_2|$, and $N_{c_1} + N_{c_2} = 0$, there is no change to the quantity in Equation 2.8.1 Therefore this value is constant under the second Reidemeister move.
(3) The third Reidemeister move preserves the orientation of each arc as well as the crossings themselves. Since there are no changes to the number of each type of crossing, we see that the quantity in Equation 2.8.1 remains unchanged.

Since the quantity

$$N_{c_1} * |c_1| + N_{c_2} * |c_2| + N_{c_3} * |c_3| + N_{c_4} * |c_4|$$

is constant under each of the three Reidemeister moves, we have proven that it constitutes a link invariant. □

We refer back to the knot diagram obtained in Figure 2.7.2, inducing an orientation on the links to obtain Figure 2.8.5. Note here that k is the same value obtained previously, the quantity $|A| - |B|$ based on the earlier definition of A and B crossings.

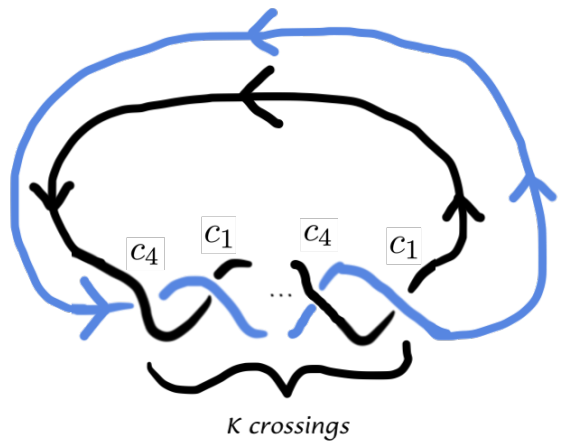


Figure 2.8.5: Link normal form, oriented.

Now, regardless of the orientations of the links, we are guaranteed that the remaining crossings will alternate between two of the new types we've defined. One of the crossing types will be c_1 or c_2, and the other will be c_3 or c_4. In Figure 2.8.5, we see that these orientations create a pairing of c_1 with c_4. Since c_1 and c_2 never appear together in the normal form, we know that the link invariants will only differ in sign, dependent on the orientation. So, depending on which crossing types are induced by the orientation of the links, we see that

$$N_{c_1} * |c_1| + N_{c_2} * |c_2| + N_{c_3} * |c_3| + N_{c_4} * |c_4| = \begin{cases} k/2 & c_1 \text{ induced} \\ -k/2 & c_2 \text{ induced} \end{cases} \tag{2.8.2}$$

Theorem 2.8.2. *For a link arising from* $\bigstar(n, 2)$*, the link is the unlink if and only if* $|A| - |B| = 0$*.*

Proof. Let $k = |A| - |B|$ where A and B are the original crossing types defined in relation to the inner polygon of $\bigstar(n, 2)$.

Then, when $k = 0$, we've shown through diagrams that we can manipulate the link into the unlink by separating its component knots. And, the value of the quantity in Equation 2.8.2 is 0.

When $k > 0$, the value of the quantity in Equation 2.8.2 will be either $k/2$ or $-k/2$, a different value than the invariant of the unlink. Since the

invariant does not change under the Reidemeister moves, the unlink is thus unattainable from links with $k > 0$. $\square$

Corollary 2.8.3. *The probability that the human knot arrangement is the unlink is*

$$\frac{\binom{n}{n/2}}{2^n}.$$

Chapter 3

Evaluating $\bigstar(7,3)$

In the previous section, we fully solved knots and links arising from $\bigstar(n,2)$. This section will focus on solving knots arising from $\bigstar(7,3)$ seen in Figure 3.0.1. These knots are significantly more complex – each has 14 crossings, with both inner crossings and outer crossings. However, this knot is still small enough to approach computationally, though there is an exponential growth in problem complexity with an increase in the number of crossings.

In some cases, we can reduce knots of type $\bigstar(7,3)$ to knots of type $\bigstar(5,2)$ and use the strategies from the previous section. Other knots we are able to validate with the knot invariant previously developed. However, sometimes we will come across knots which our invariant does not distinguish from the unknot, and we cannot simply see that these knots are unknotted. These knots we classify for now as unknown.

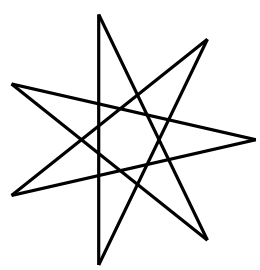


Figure 3.0.1: $\bigstar(7,3)$

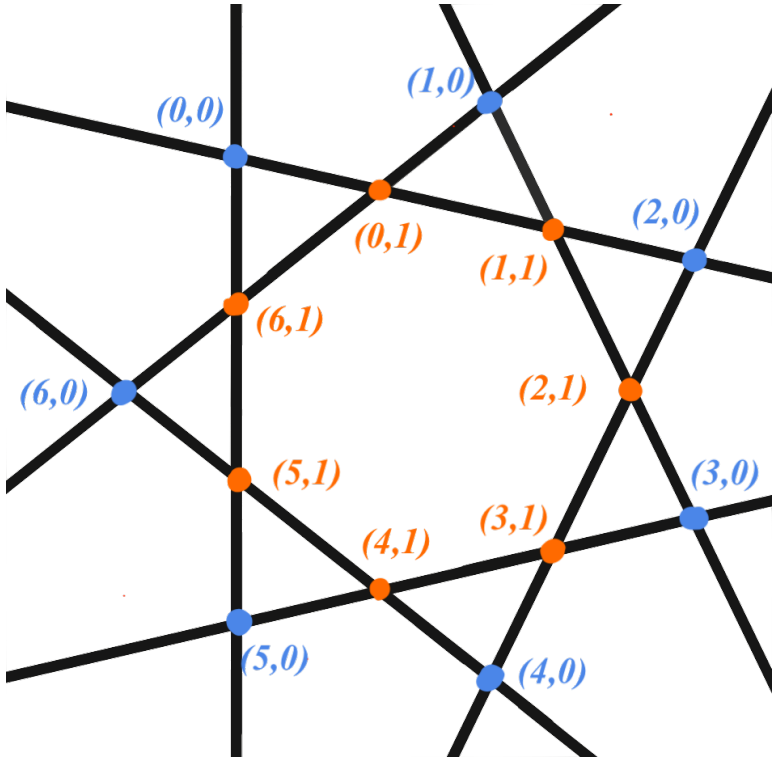


Figure 3.1.1: An enumeration of the crossings of $\bigstar 7, 3$.

3.1 Enumerating the Knot

To enumerate the knot in a useful way, we use a modular approach. We divide the knot into 7 identical sections, corresponding to the area immediately surrounding each of the 7 vertices. A labeling of this enumeration can be found in Figure 3.1.1.

3.2 Reduction to $\bigstar(5, 2)$

Looking at $\bigstar(7, 3)$, we see immediately that there are ways to reduce that star to the star we had in the previous chapter, $\bigstar(5, 2)$.

3.2.1 An Example Reduction

Consider the following knot diagram:

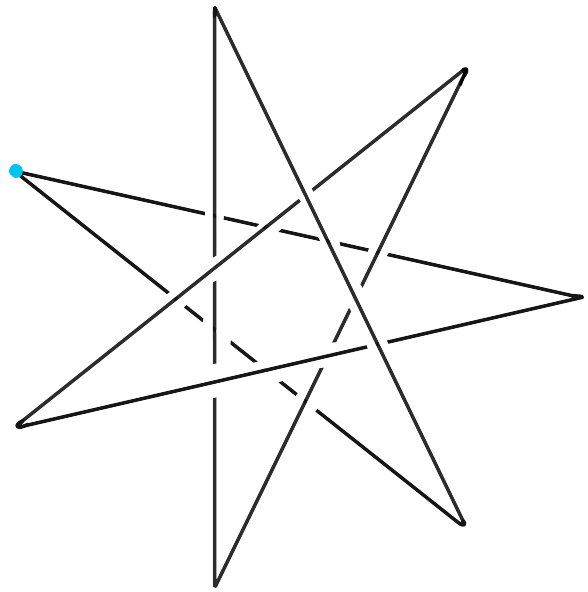


Figure 3.2.1: $\bigstar(7,3)$ with vertex fixed, cases 1-4.

In this case, we can choose an outer vertex and apply some Reidemeister moves to manipulate the knot. In fact, we'll be able to transform this knot to a knot of the form $\bigstar 5,2$.

Looking at the circled crossings in Figure 3.2.2, we see that each of these crossings allows us to move our fixed vertex under the rest of the knot.

Now, all that's left to do is 'twist' the fixed vertex to remove a final extraneous crossing. Then, we are left with a knot of the form $\bigstar(5,2)$.

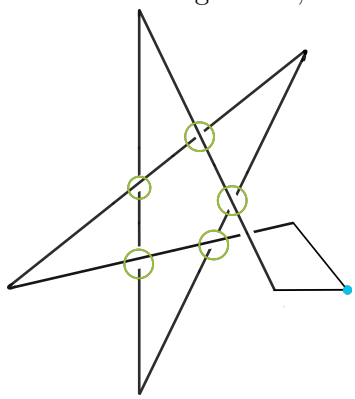


Figure 3.2.3: $\bigstar(7,3)$ after moving the fixed vertex across the knot.

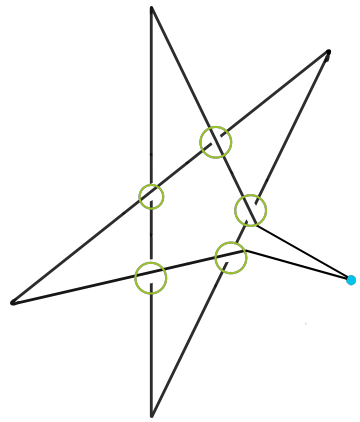


Figure 3.2.4: $\bigstar(5,2)$.

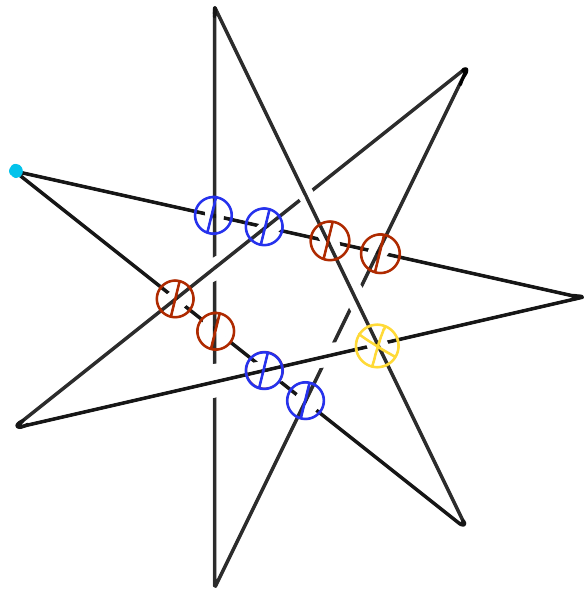


Figure 3.2.2: $\bigstar(7,3)$ with crossings to check circled in red and blue

Looking back at the original knot, let's identify those crossings which remain after the reduction. By looking at these crossings, circled in Figure 3.2.5, we can determine if the knot is the unknot by counting As and Bs when the knot is reducible.

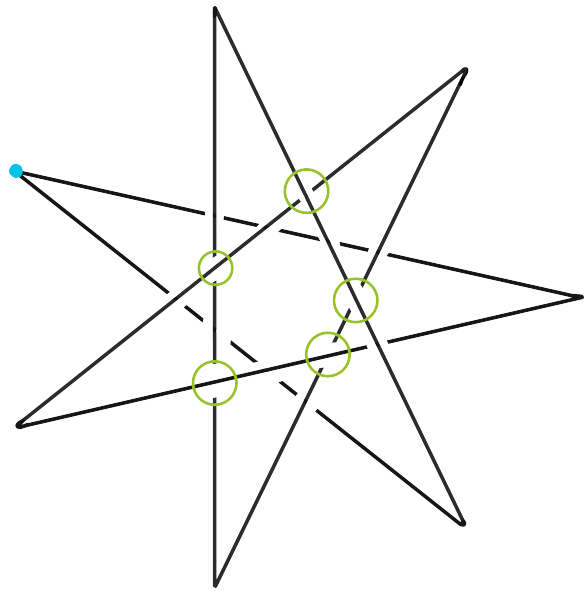


Figure 3.2.5: $\bigstar(7,3)$ with the remaining 5 crossings after reduction

In the next subsection, we will dive deeper into what conditions allow us to reduce a knot of the form $\bigstar(7,3)$ to a $\bigstar(5,2)$ knot. We will also more rigorously outline the steps of reduction.

3.2.2 Detailing the Reduction Process

First, fix an outer vertex. This outer vertex has two arcs emanating from it. An arc spans from undercrossing to undercrossing. Consider the first arc. Call the union of the arcs spanning from outer-vertex to opposite outer vertex an arm. There are 4 crossings along this arm. Enumerate them in order. If the first two are both A crossings or B crossings, then the second two must be the opposite type of crossing. If this is true for both of the arms emanating from this vertex, then we are able to 'unwind' these segments of the knot and reduce the $\bigstar(7,3)$ knot into an instance of the $\bigstar(5,2)$ knot.

Additionally, we can unravel the knot when the four crossings on the vertex-side of the inner polygon place the arm under the rest of the knot, or going over the rest of the knot, and the same is true for the four crossings opposite the chosen vertex. The precise list of cases where we can reduce is given by Table 3.2.1.

We introduce a notation to more formally discuss these crossings. Fix an outer vertex as V_0. Then, we denote the outermost crossings as 0-crossings

Case	First arm	Second arm
1	$[A, A, B, B]$	$[A, A, B, B]$
2	$[A, A, B, B]$	$[B, B, A, A]$
3	$[B, B, A, A]$	$[A, A, B, B]$
4	$[B, B, A, A]$	$[B, B, A, A]$
5	$[A, A, A, A]$	$[B, B, B, B]$
6	$[B, B, B, B]$	$[A, A, A, A]$

Table 3.2.1: Cases when $\bigstar(7, 3)$ can be reduced to $\bigstar(5, 2)$.

Crossing types	Case
$[A, B, A, B]$	1
$[A, B, B, A]$	2
$[B, A, A, B]$	3
$[B, A, B, A]$	4
$[A, A, B, B]$	5
$[B, B, A, A]$	6

Table 3.2.2: Final check for reduction with corresponding cases.

and the innermost crossings as 1-crossings. Label these crossings starting with the crossing immediately counter-clockwise from V_0. Then, each of these crossings can be referred to by (i, j), where i refers to the enumeration of the crossing and j refers to the type of crossing, either 0 or 1. An example of the labeling of a single outermost vertex is given in Figure 3.2.6, and is consistent with the labeling in 3.1.1.

To evaluate whether a knot can be reduced, we check the following equalities, when considering the arms emanating from V_i:

$$(i, 0) = (i - 1, 1) \tag{3.2.1}$$
$$(i - 2, 1) = (i - 2, 0) \tag{3.2.2}$$
$$(i + 1, 0) = (i + 1, 1) \tag{3.2.3}$$
$$(i + 2, 1) = (i + 3, 0). \tag{3.2.4}$$

If all of these equalities are satisfied, then we need only check that $[(i, 0), (i - 2, 0), (i + 1, 0), (i + 3, 0)]$ is one of the lists enumerated in Table 3.2.2. This will ensure that the pairings along the arms alternate in such a way that we can reduce.

A graphical representation of the first four cases of this reduction is given

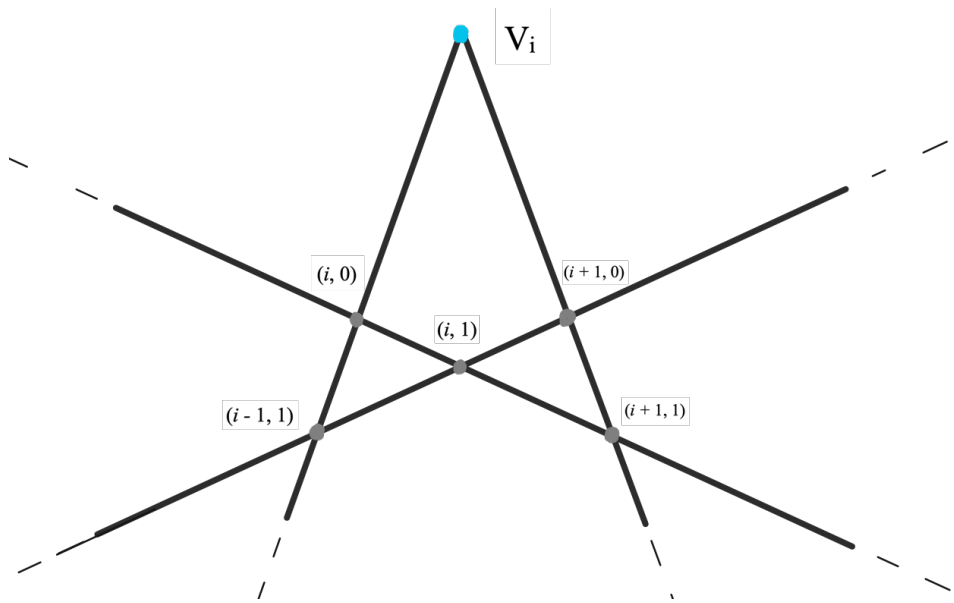


Figure 3.2.6: A segment of the labeling of $\bigstar(7,3)$, including outer vertices

in Figures 3.2.1 – 3.2.4. A representation of the final two cases is given in Figures 3.2.7 – 3.2.12.

Heuristic 3.2.1. *The Over-Counting Heuristic. An upper bound for the number of permutations of* $\bigstar(7,3)$ *which can be reduced is* 2^{10}.

Proof sketch. Fix a vertex. Then, check the first four equalities. How often is each equality satisfied? Each equality can either be true or false, so each is satisfied by 1/2 of all the candidate knots. So all of the equalities are satisfied by 1/16 of the knots. This gives us 1/64 of knots which satisfy our inequalities for a given vertex. Then, since there are 7 vertices, we multiply this result by 7 to receive an estimate (up to double-counting) of the number of knots we can reduce with this strategy. This gives us a proportion 7/64 knots which could potentially be reduced with the strategy. □

So, it seems that we can relatively frequently reduce $\bigstar(7,3)$ to $\bigstar(5,2)$. We'll find in practice that the actual number of knots we can reduce is about 1/4 of the overestimated total, with the results in Table 3.3.1.

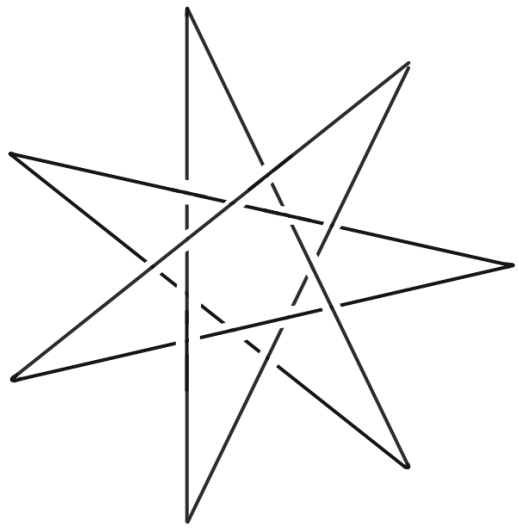


Figure 3.2.7: Cases 5 and 6 for reduction.

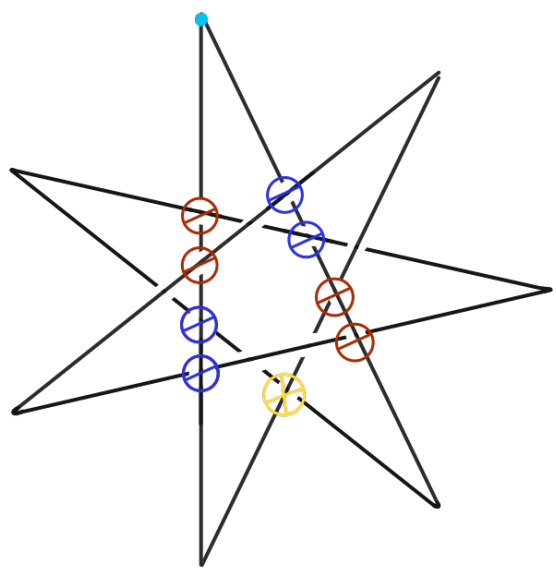


Figure 3.2.8: Cases 5 and 6 with eliminated crossings highlighted.

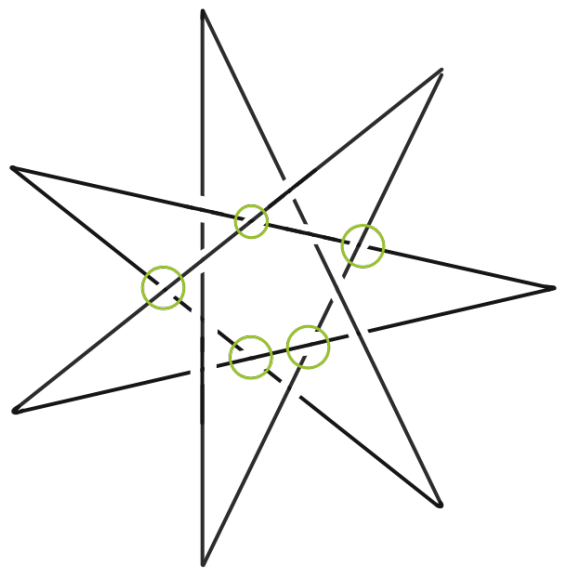


Figure 3.2.9: Cases 5 and 6 with remaining crossings highlighted.

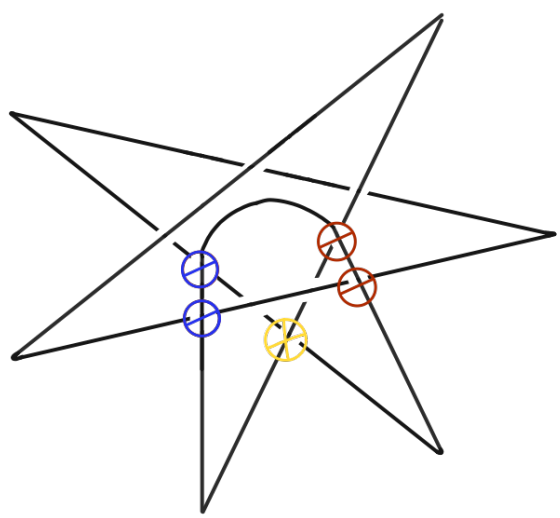


Figure 3.2.10: First step in reducing cases 5 and 6.

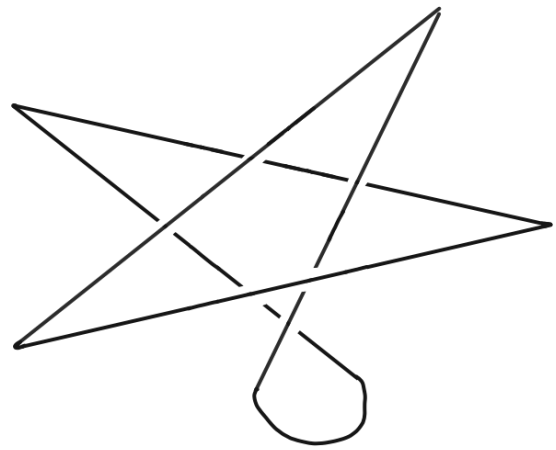


Figure 3.2.11: Second step in reducing cases 5 and 6.

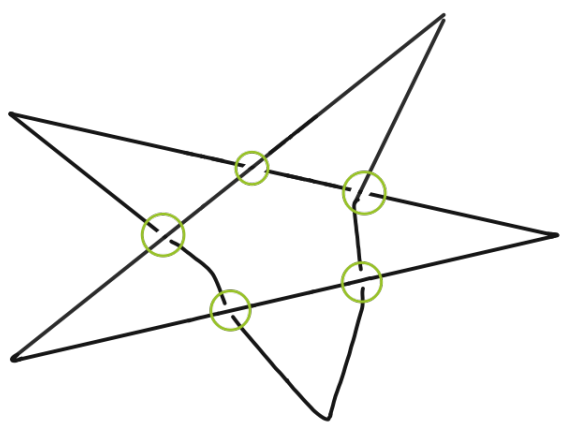


Figure 3.2.12: Reduction to $\bigstar(5, 2)$ complete.

3.3 Evaluating Result

Preliminary evaluations of the resulting output reveal that the majority of these knots are definitively knotted. A small table of results is seen in Table 3.3.1.

Result	Count
Unknotted	246
Knotted	13568
Unknown	2570

Table 3.3.1: Program output analyzed

The program was able to reduce 396 of the knots, about a factor of 4 off of Heuristic 3.2.1. Additionally, as a means of validation, we ran the prime test on knots which were reduced in order to ensure that the two solving methods were consistent with one another. They were. Additionally, knots which were found to be knotted upon reduction were also exclusively differentiated by the primes 3 and 5, depending on their $k = |A| - |B|$ value, agreeing with the finding in Theorem 2.3.1 that there are n^2 colorings when $k \equiv 0 \mod p$.

We can also take a look at the primes which ended up differentiating knots from the unknot. Interestingly, some knots are differentiated by multiple primes. We show the counts of specific groupings of those primes in Table 3.3.2

The individual primes which appeared as differentiating knots from the unknot are another compelling object of study. Counting instances when these primes were also part of a pairing, we see the counts of primes appearing in the results in Table 3.3.3. Notably, 53 only appeared as a part of a pairing, paired with 5. All the other primes which appeared in a pairing also appeared as the only prime to differentiate at least one knot. It seems as though as the prime increases, it differentiates fewer knots, although there are some notable exceptions: 13, for one, seems to have a high number of differentiated knots, while 11 has a particularly low number of differentiated knots. It could be fruitful to investigate the cause of these discrepancies.

Future work on these knots in particular could take aim at the relation that differentiating primes have with the knot that they differentiated, either as a knot itself or as its matrix representation.

Prime groups	Count
$(3, 13)$	140
$(3, 23)$	28
$(3, 2)$	168
$(3, 5, 11)$	28
$(3, 5)$	476
$(3, 7)$	28
$(5, 17)$	28
$(5, 53)$	28
$(5, 7)$	168
$(7, 11)$	28
$(7, 13)$	140
$(7, 19)$	28

Table 3.3.2: Counts of prime pairs differentiating knots

Prime	Count
(3)	6,258
(5)	2,464
(7)	2,366
(11)	420
(13)	1498
(17)	588
(19)	168
(23)	364
(29)	170
(31)	280
(37)	84
(41)	28
(53)	28
(61)	84
(73)	56
(97)	28

Table 3.3.3: Counts of primes appearing in results

Appendices

Appendix A

Implementing the Reduction and Invariant in Sage

We closely follow the procedures outlined. All code is written in Sage, an extension of Python. We use the Python libraries `numpy`, `scipy`, and `math`.

A.1 Coding the reduction procedure

We follow the procedure outlined in Section 3.2.2.

```
def reduction_procedure(crossings_0, crossings_1):
    acceptable_list = [[1,0,1,0],[0,1,1,0],[1,0,0,1],[0,1,0,1],[0,0,1,1],[1,1,0,0]]
    for i in range(7):
        #for each outer vertex:
        #n.b. 0 corresponds to A in all cases.
        if(reduction_check_crossings(crossings_0, crossings_1, i)):
            #crossings might allow reduction
            if get_key_crossings(crossings_0, crossings_1,i) in acceptable_list:
                #crossings definitely allow reduction
                if (check_remaining_crossings(crossings_0, crossings_1,i)):
                    #then we have a knot which is 5,2 rearranged, and
                    # 2 or 3 of the crossings are of type A.
                    # therefore we have found an unknot
                    return REDUCED_UNKNOT
                else:
                    return REDUCED_KNOT
        else:
```

```
        return UNABLE_TO_REDUCE
```

The helper functions called above mirror the procedures in the previous section as well, where `crossings_0` and `crossings_1` refer to outer and inner crossings, respectively. Inner here refers to those crossings which lie on the inner polygon of $\bigstar(7, 3)$.

```
def reduction_check_crossings(crossings_0, crossings_1, i):
    return ((crossings_0[i % n]      == crossings_1[(i-1) % n]) and
            (crossings_0[(i-2 % n)] == crossings_1[(i-2) % n]) and
            (crossings_0[(i+1) % n] == crossings_1[(i+1) % n]) and
            (crossings_0[(i+3) % n] == crossings_1[(i+2) % n]))

def get_key_crossings(crossings_0, crossings_1, i):
    return ([crossings_0[i],
             crossings_0[(i-2) % n],
             crossings_0[(i+1) % n],
             crossings_0[(i+3) % n]])

def check_remaining_crossings(crossings_0, crossings_1, i):
    sum_crossings = (crossings_1[i] +
            crossings_1[(i+4) % n] +
            crossings_1[(i+3) % n] +
            crossings_0[(i+2) % n] +
            crossings_0[(i-1) % n])
    return sum_crossings == 2 or sum_crossings == 3
```

A.2 Constructing Knot Incidence Matrix

We will make use of knot invariant discussed in 2.4.1 as well as modular linear algebra to calculate whether or knot is equivalent under our invariant to the unknot. To do this, we want to find the dimension of the nullspace of the incidence matrix of the arcs of the knot.

To construct a knot incidence matrix for each knot efficiently, we begin by constructing a master knot incidence matrix from which we sample rows. The largest matrix, M, is an incidence matrix of the involvements of arcs in each crossing. Each row corresponds to a crossing, and each column corresponds to an arc. Values will be assigned based on the involvement of each arc with the enumerated crossings, either 1 or -1. Arcs along different arms are assigned opposite values.

Then, we construct 4 other matrixes to differentiate crossings as A or B crossings. We refer to these matrices as $M_{0A}, M_{0B}, M_{1A}, M_{1B}$, and each will identify the values of the over arc for a given crossing to satisfy the equation of the invariant, equation 2.4.1. Here, 0 and 1 refer to outer and inner crossings respectively, where inner crossings are those which lie on the inner polygon of $\bigstar(7,3)$.

```
def setup_master_matrices():
    I = matrix.identity(n)
    O = matrix.zero(n)

    shift = np.concatenate([np.array([n-1]),np.arange(n-1)])
    L = matrix.zero(n)
    for i in range(n):
        L[i] = I[shift[i]]
    R = L.transpose()

    M = block_matrix([[I-L, I, -I, O], [O, R, -I, I-R]], subdivide=False)
    M_0A = block_matrix([[I, -I, O, O]], subdivide=False)
    M_0B = block_matrix([[L, O, -I, O]], subdivide=False)
    M_1A = block_matrix([[O, R, O, -I]], subdivide=False)
    M_1B = block_matrix([[O, O, I, -R]], subdivide=False)
    return (M, M_0A, M_0B, M_1A, M_1B)
```

A.3 Assigning crossings

To assign crossings for a particular knot, we look through the crossing arrays, which will be arrays consisting of entirely 0s and 1s. If we see a zero, we choose a row from the A crossing matrix. If we see a 1, we choose a row from the B crossing matrix. We pass `crossings_0` and `crossings_1` as arguments through each procedure. Since we are only reading from these arguments, we do not need to worry about pass-by-value versus pass-by-reference. We build up M_0X and M_1X separately and then combine them with M to construct an incidence matrix for a particular crossing assignment.

```
def setup_m0x():
    M_0X = matrix.zero(n,4*n)
    for k in range(n):
        if (crossings_0[k] == 0):
            M_0X[k] = M_0A[k]
```

```
            #print(M_0X[k])
        elif (crossings_0[k] == 1):
            M_0X[k] = M_0B[k]
        else:
            print("error")
    return M_0X

def setup_m1x():
    M_1X = matrix.zero(n,4*n)
    for k in range(n):
        if (crossings_1[k] == 0):
            M_1X[k] = M_1A[k]
        elif (crossings_1[k] == 1):
            M_1X[k] = M_1B[k]
        else:
            print("error")
    return M_1X
```

Then, to finally construct our particular knot incidence matrix for a given iteration, we combine matrices.

```
def construct_knot_matrix(M, M_0X, M_1X):
    return block_matrix([[M], [M_0X], [M_1X]], subdivide=False)
```

A.4 Retrieving every crossing assignment

Since there are 14 crossings for this class of knot, we must run through $2^{14} = 16384$ iterations of knots. Now, some of these will be knots isomorphic to one another under certain translations, but for the purposes of this program, we need not eliminate such redundancies. We generate crossing assignments by converting integers to binary and then placing each binary digit into arrays of length 7, one array for `crossings_0` and one for `crossings_1`.

```
def list_from_integer(repetition):
    first_list = [int(d) for d in str(bin(repetition))[2:]]
    while len(first_list) < 14:
        first_list = [0] + first_list
    return (first_list[0:7], first_list[7:])
```

This returns a tuple of lists.

A.5 Looping through every crossing assignment

Here, we put all of the pieces together and run a large loop to iterate over each of the possible crossing arrangements. The code is found below. **Code:**

```
n = 7

#constants for reduction
UNABLE_TO_REDUCE = 0
REDUCED_KNOT = 1
REDUCED_UNKNOT = 2

#functions to help with printing
def reduce_to_knot():
    return (1,0,0,1)

def reduce_to_unknot():
    return(0,1,0,1)

def prime_knot():
    return (1,0,0,0)

def start_conditions():
    return (0,0,1,0)

#primes up to 100
primes = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]

#file to write output to
output = open('output.txt', 'w')

#print header
print("Iteration \t inner crossings \t outer crossings" +
        " \t unknotted \t knotted \t unknown \t primes_knotted \t reduced",
         file=output)


#set up master matrix
(M, M_0A, M_0B, M_1A, M_1B) = setup_master_matrices()
```

```
count = 0

#run over all 2^14 crossing assignments
for i in range(16384):

    #each knot begins unknown

    (knotted, unknotted, unknown, reduced) = start_conditions()
    is_reduced = False
    #keep track of primes which differentiate the knot
    #under our invariant
    primes_knotted = []

    #set up the knot matrix based on crossing lists for this iteration
    (crossings_0, crossings_1) = list_from_integer(i)

    M_0X = setup_m0x()
    M_1X = setup_m1x()

    knot_matrix = construct_knot_matrix(M, M_0X, M_1X)

    #Reduction tests
    reduction_result = reduction_procedure(crossings_0, crossings_1)

    if(reduction_result == REDUCED_UNKNOT):
        #this case reduces to the unknot
        (knotted, unknotted, unknown, reduced) = reduce_to_unknot()
        is_reduced = True
    elif(reduction_result == REDUCED_KNOT):
        (knotted, unknotted, unknown, reduced) = reduce_to_knot()
        is_reduced = True
    #Invariant tests with field

    print("checking primes" + str(i))
    #run over all primes
    for p in primes:
        X = MatrixSpace(GF(p), 4*n, 4*n)
        if X(knot_matrix).nullity() > 1:
            #this case is definitely a knot
            primes_knotted = primes_knotted + [p]
```

```
            (knotted, unknotted, unknown, reduced) = prime_knot()
            if is_reduced:
                reduced = 1
    print("" + str(i) + "\t " + str(crossings_1) + "\t " + str(crossings_0) + "\t "
          + str(unknotted) + "\t " + str(knotted) + "\t " + str(unknown) + "\t "
          + str(primes_knotted) + "\t " + str(reduced), file=output)

output.close()
```

Bibliography

[1] Alexander, J. W. Topological invariants of knots and links. *Transactions of the American Mathematical Society 30*, 2 (1928), 275–306.

[2] Benedetti, F. (1) (PDF) Studies of global and local entanglements of individual protein chains using the concept of knotoids.

[3] Jones, V. F. R. A polynomial invariant for knots via von neumann algebras. *Bull. Amer. Math. Soc. (N.S.) 12*, 1 (01 1985), 103–111.

[4] Kosniowski, C. *A First Course in Algebraic Topology.* Cambridge University Press, 1980.

[5] Reidemeister, K. Elementare begrndung der knotentheorie. *Abhandlungen aus dem Mathematischen Seminar der Universitt Hamburg 5*, 1 (dec 1927), 24–32.