Conjectures on Khovanov and Knot Floer Homology and an Algorithm for the Jones Polynomial

Ryan Maguire

February 17, 2022

Corrections

A bug was discovered in the Knot Floer Homology (KFH) code that led to false negatives. The conjecture as presented in this talk for KFH and Legendrian simple knots is false and counterexamples have been found. For Khovanov homology the conjecture still stands as of this correction (written 2023/02/27).

The wording of the theorem about Gauss codes has been corrected, and references given.

The remainder of these slides are unchanged from the original talk.

Outline

- Gauss Code
- The Kauffman Bracket and Jones Polynomial
- Khovanov Homology
- An Algorithm for the Jones Polynomial
- Conjectures on Khovanov and Knot Floer Homology

Gauss Code

Take a knot, orient it, and label the crossings from 0 to $N-1$. Starting at the 0 crossing, travel along the knot and when you come to a crossing, record the crossing number and whether you're on the *over* strand or the *under* strand. The string of length $2N$ you've obtained is the *Gauss code* of the knot.

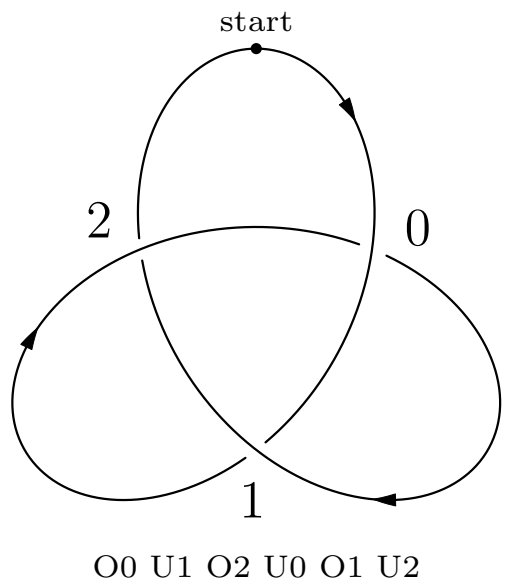

Figure: Gauss Code for the Right-Handed Trefoil

Gauss Code

Problem: Different knots can have the same Gauss code.[1] Take the left-handed trefoil, similar orientation and labelling scheme as before, but different starting point. The resulting code is the same as before. The left and right handed Trefoils are different (Their Jones polynomials are different).[2]

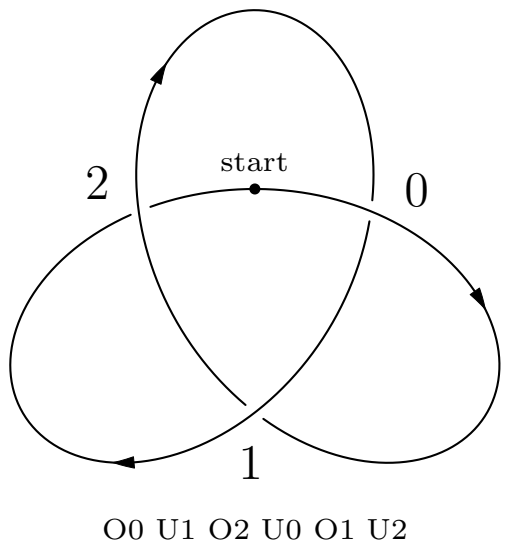

Figure: Gauss Code for the Left-Handed Trefoil

[1] It bothers me that there's only *one* trefoil knot in every knot table.

[2] Did it really take until the 1980's to know these are different?

Gauss Code

Theorem

If two classical prime knots have the same Gauss code, either they are isomorphic, or they are mirrors of each other.

By *classical* it is meant that the knots are not virtual (discussed in a few slides). Dowker and Thistlethwaite proved that DT code uniquely specifies a prime knot up to reflection [2], and DT codes contain the same information as Gauss codes. Indeed, there are algorithms to go from one to the other [1].

We have an example of two distinct knots having the same Gauss code, the left and right handed trefoils. How can we describe knots while distinguishing mirrors?

Gauss Code

Solution: Sign the crossings. At every crossing, rotate your head until the crossing looks like one of the ones below. Call the one on the left a *negative crossing* and the one on the right *positive*.

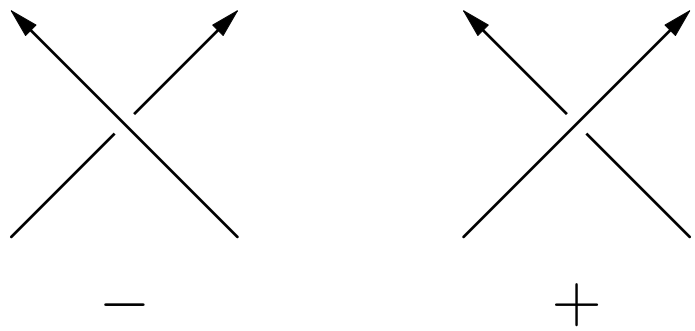

Figure: Crossing Signs

Gauss Code

By also recording the sign, we get *extended Gauss code*.

Theorem

If two classical prime knots have the same extended Gauss code, they are isomorphic.

Since unsigned codes distinguish knots up to mirror equivalence, and since signing the code detects mirrors, the result is almost immediate.

Gauss Code

Below is the extended Gauss code for the right-handed trefoil.

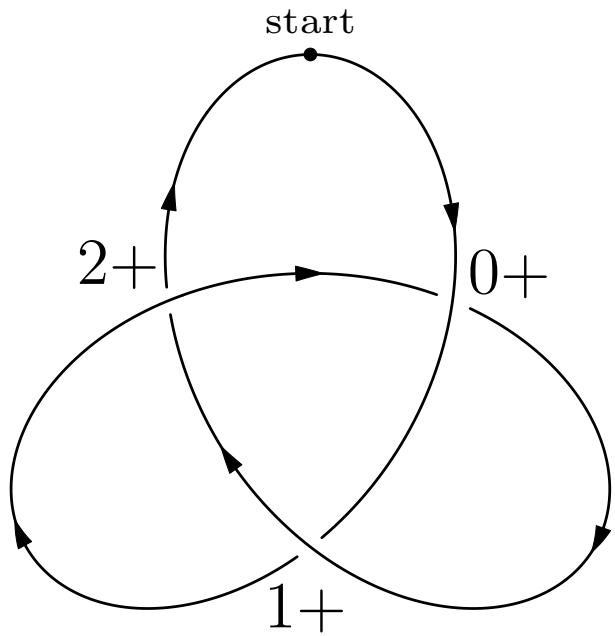

Figure: Extended Gauss Code for the Right-Handed Trefoil

Gauss Code

The rules:

1. Gauss code is a finite sequence of length $2N$ of ordered triples.
2. The ordered triples are of the form (s, n, t) with $s \in \{-1, +1\}$, $t \in \{O, U\}$, and $0 \leq n \leq N - 1$. (s for *sign*, t for *type*).
3. Every integer $0 \leq n \leq N - 1$ occurs exactly twice in the code, once with O and once with U. The sign does not change.

Gauss Code

The three Reidemeister moves translate to fairly simple operations on Gauss code.[3] We will be discussing an algorithm for the Jones polynomial that is exponential in crossings and it would be nice if there were a simpler knot than either of the trefoils. And there is!

$$O0\ O1\ U0\ U1 \tag{1}$$

You'll find no Reidemeister moves can reduce this to the unknot.

[3] HW what are they?

Gauss Code

If you try to draw the knot from this code, you'll get the following. There's this *fake* crossing.

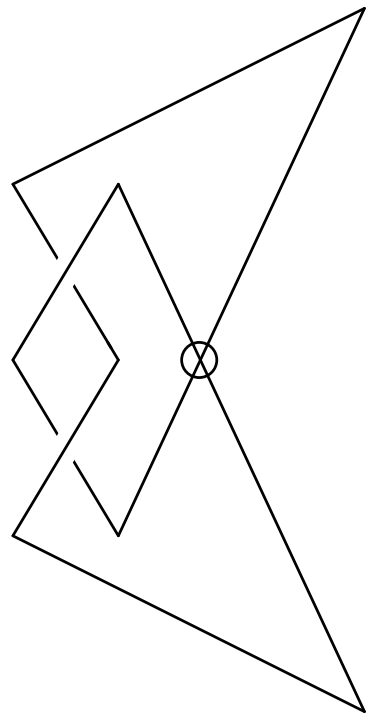

Figure: The Chain Link Fence Knot

Gauss Code

The graph theorist in you knows that we really want to draw this on a torus.[4]

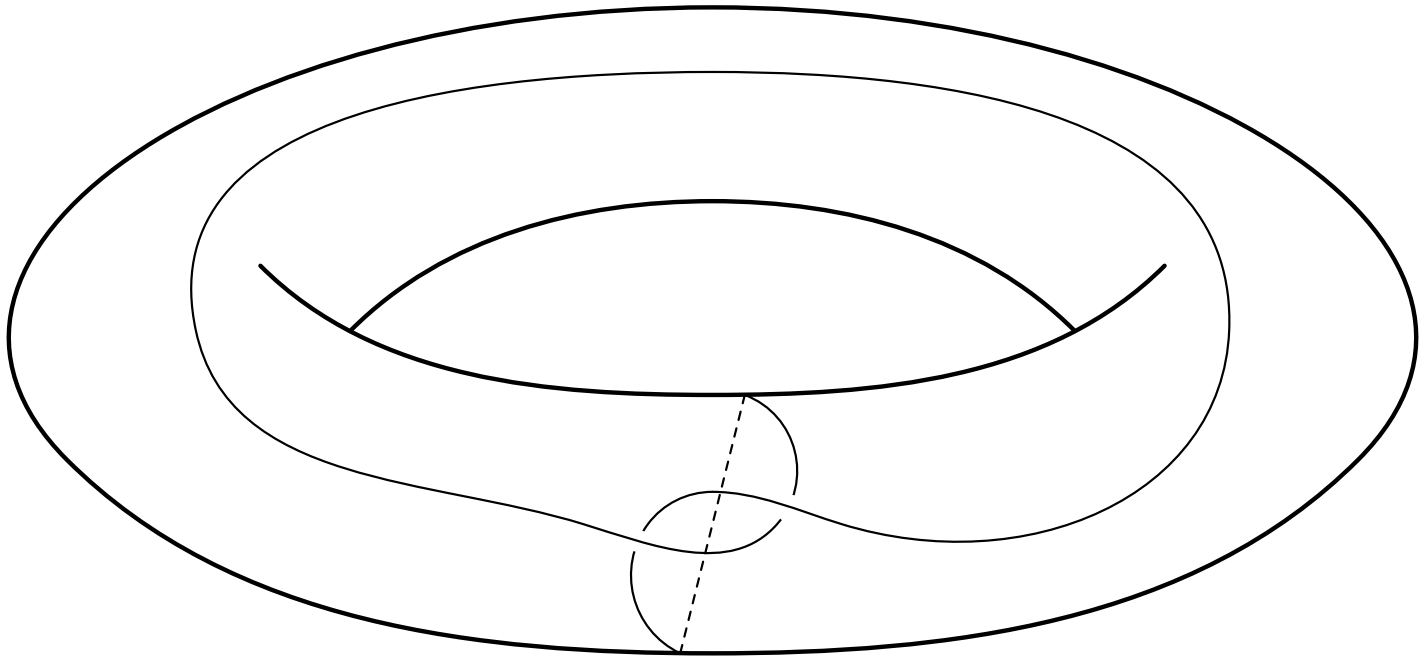

Figure: The Chain Link Fence Knot on a Torus

[4] Ever try and draw K_5 without crossings?

Gauss Code

But what shall we name it (don't read the figures label)? Let's draw it on a flat torus.

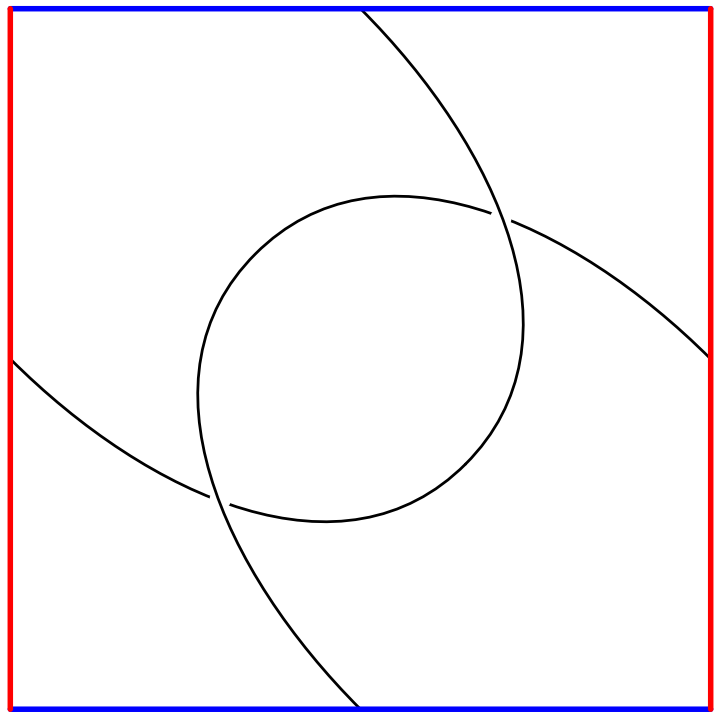

Figure: The Chain Link Fence Knot on a Flat Torus

Gauss Code

And then let's lift that drawing to the universal cover.

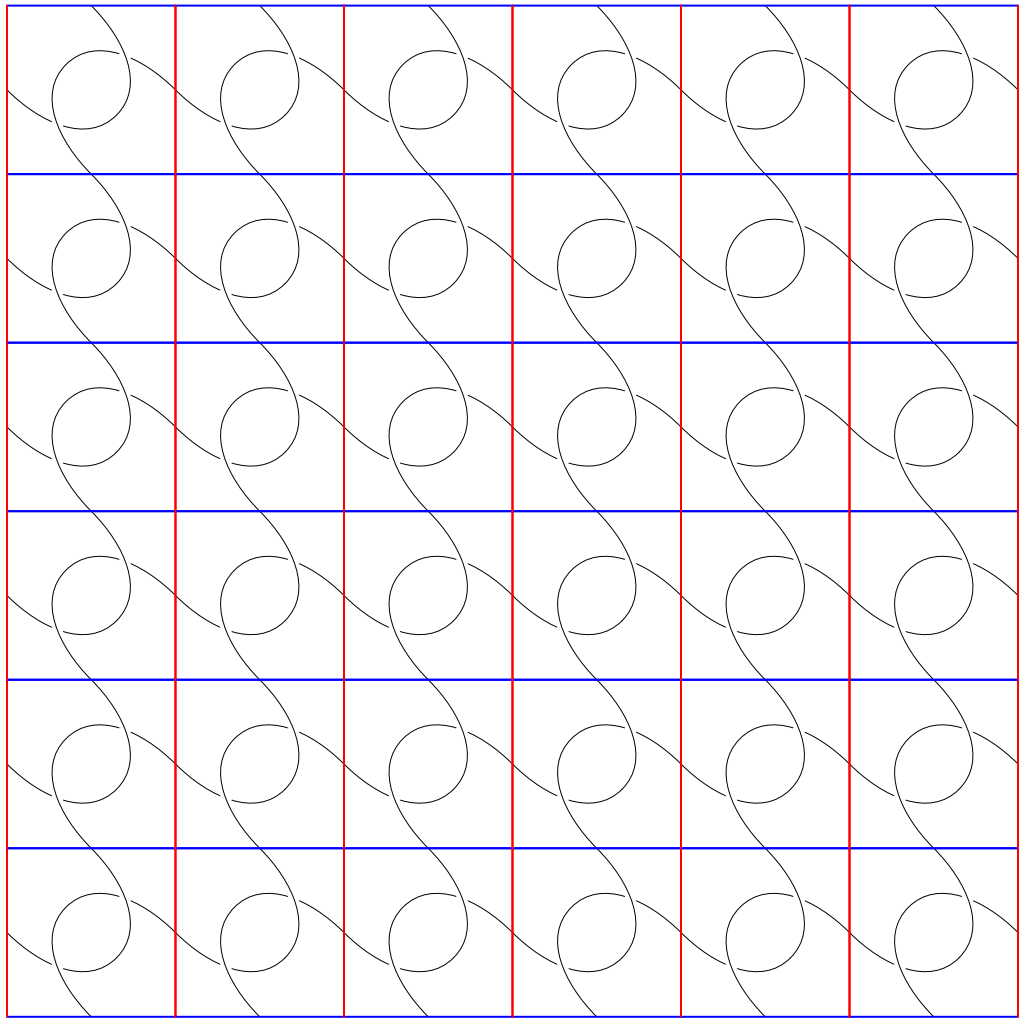

Figure: Lift of the Chain Link Fence Knot to $\mathbb{R}^2$

Gauss Code

Looks like a chain-link fence. Let's call it that. This is the simplest non-trivial *virtual knot*. A classical knot can be thought of as a smooth embedding of $\mathbb{S}^1$ into $\mathbb{S}^2 \times \mathbb{R}$. Virtual knots are smooth embeddings of $\mathbb{S}^1$ into $M \times \mathbb{R}$ where M is a compact surface. The *virtual genus* of a virtual knot is the smallest possible genus of M. Classical knots have virtual genus 0. The chain-link fence knot has virtual genus 1.

Gauss Code

We can detect the virtual genus from extended Gauss code. Take the right-handed trefoil and thicken it. Start anywhere you'd like, but place your finger on the *left* strand and start walking forward. When you encounter a crossing, go left! Eventually you'll end up back where you started. Keep doing this until you've traversed all of the strands, keeping track of the total number of cycles.

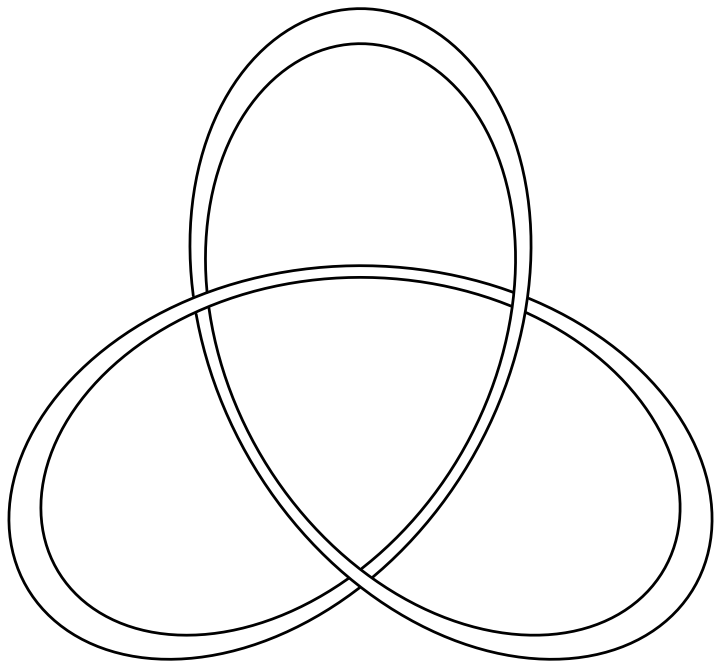

Figure: Framed Left-Handed Trefoil

Gauss Code

To conclude, use the following:

$$V - E + F = 2 - 2g \tag{2}$$

You've just computed F! V is the number of crossings, and E is always $2V$ (the knot graph is a four-valent multigraph. By the hand-shaking theorem, $E = 4V/2 = 2V$). So:

$$g = \frac{V - F + 2}{2} \tag{3}$$

We don't care about the virtual genus right now. But we do care about the idea. We will modify it later to get the Jones polynomial.

The Kauffman Bracket and Jones Polynomial

The Kauffman Bracket is defined recursively in terms of smoothings of a crossing. Given a crossing, there are two ways to make it *go away*. We will label these the 0 and 1 smoothings, see below.

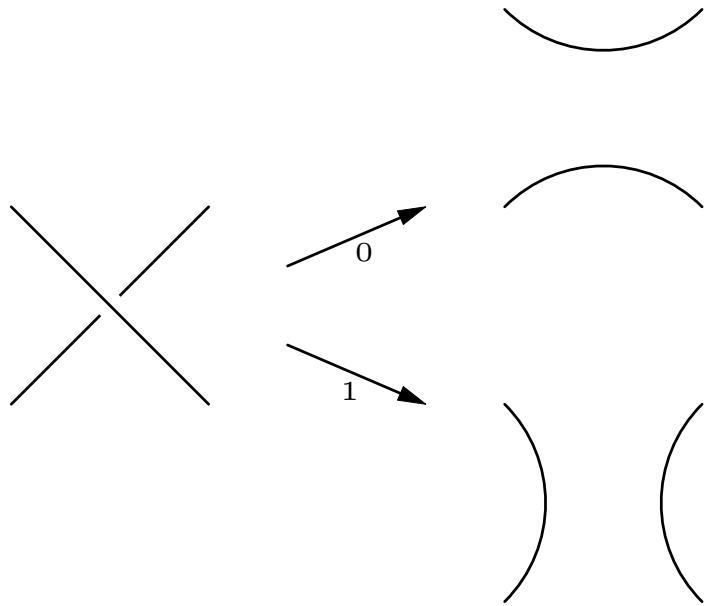

Figure: Resolving a Crossing

The Kauffman Bracket and Jones Polynomial

Given a knot with N crossings, with crossings labeled 0 to $N-1$, and any integer $0 \leq n \leq 2^N - 1$, there is a unique resolution of all crossings corresponding to n. Write n in binary. The value of the k^{th} bit tells us how we are supposed to smooth the k^{th} crossing.

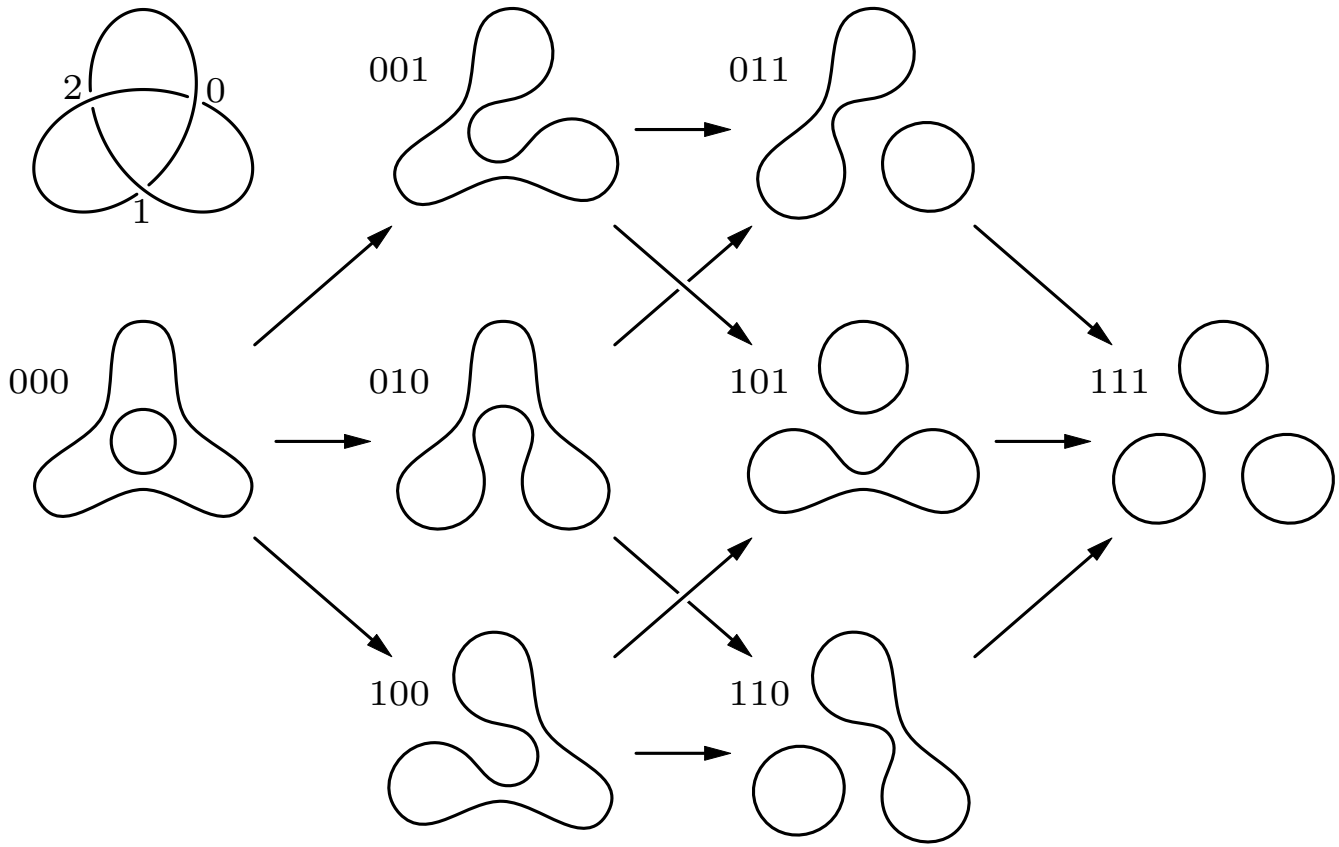

Figure: Cube of Resolutions for the Left-Handed Trefoil

The Kauffman Bracket and the Jones Polynomial

The following took too long to make.

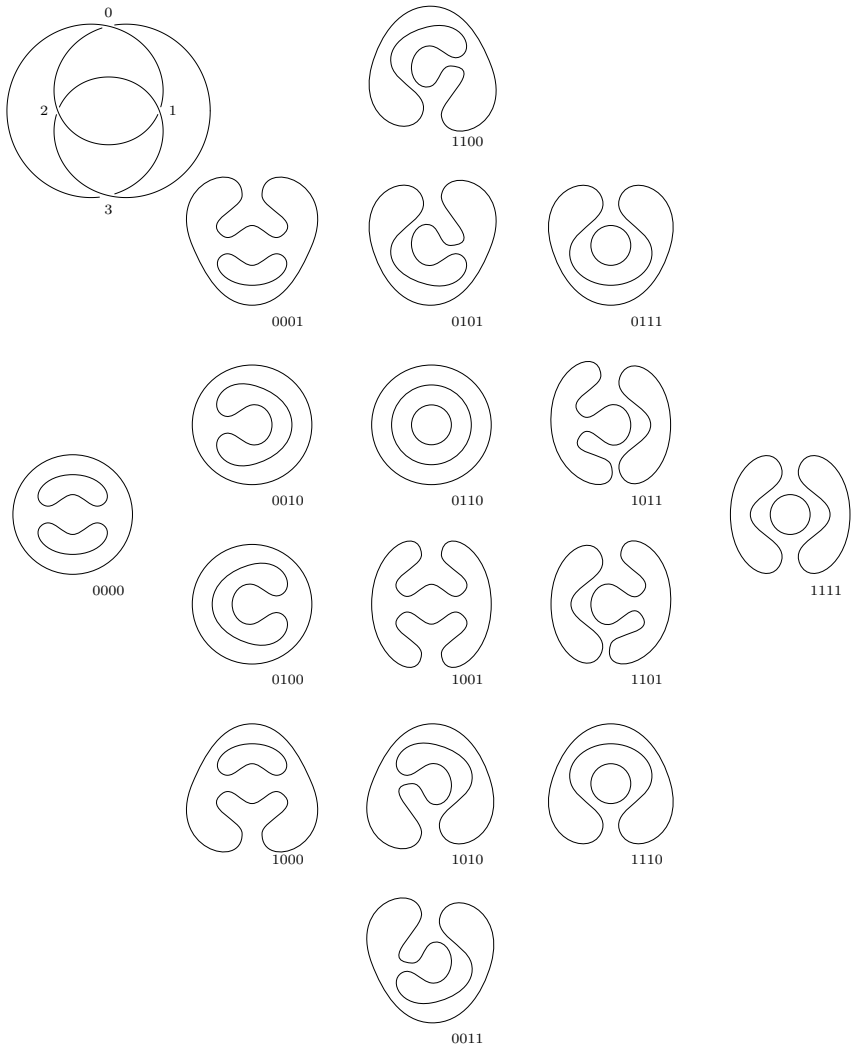

Figure: Cube of Resolutions for the Figure-Eight

The Kauffman bracket is defined recursively as follows:

$$\langle \emptyset \rangle = 1 \tag{4}$$

$$\langle L \sqcup \mathbb{S}^1 \rangle = (q + q^{-1})\langle L \rangle \tag{5}$$

$$\langle L \rangle = \langle L_{n,0} \rangle - q\langle L_{n,1} \rangle \tag{6}$$

where $L_{n,0}$ and $L_{n,1}$ are the links obtained from the 0 and 1 smoothings of L at the n^{th} crossing, respectively. The notation $L \sqcup \mathbb{S}^1$ means the disjoint union of L with an unknot. Hence the Kaufmann bracket of the unknot is $q + q^{-1}$.

Theorem

The Kauffman bracket is invariant under Reidemeister II and III moves.

It is not invariant under Reidemeister I, so we have a *framed knot* invariant.

The Kauffman Bracket and the Jones Polynomial

If you have something that is invariant under Reidemeister II and III, you should try to introduce the *writhe*[5] into the problem since only Reidemeister I changes the writhe of a diagram. The Jones polynomial does exactly this:

$$J(L) = (-1)^{N_-} q^{N_- - 2N_+} \langle L \rangle \tag{7}$$

$N_\pm$ being the number of positive and negative crossings, respectively.

Theorem

The Jones polynomial is a knot invariant.

[5] Sum of the signs of the crossings

Khovanov Homology

You can get a homology theory out of this. Khovanov homology is the "categorification" of the Jones polynomial, the graded Euler characteristic of which gives you the Jones polynomial. I won't be discussing heavy details about Khovanov homology, only some of the technical difficulties involved in its computation. The recursive definition of the chain complex is as follows:

$$[[\emptyset]] = 0 \to \mathbb{Z} \to 0 \tag{8}$$

$$[[L \sqcup \mathbb{S}^1]] = V \otimes [[L]] \tag{9}$$

$$[[L]] = \mathcal{F}\big(0 \to [[L_{n,0}]] \to [[L_{n,1}]]\{1\} \to 0\big) \tag{10}$$

V is a graded vector space, $\{\ell\}$ is the *degree shift* operation, and $\mathcal{F}$ is the flatten operation of graded vector spaces. Like the Kauffman bracket, we need the writhe to a get a true knot invariant. The chain complex is $C(L) = [[L]][N_-]\{N_- - 2N_+\}$, $[\ell]$ is the height shift operation on chain complexes. Khovanov homology is the resulting homology from this (chain maps are not defined in this talk).

Khovanov Homology

The naïve recursive algorithm for Khovanov homology is exponential in both time and space. Most algorithm talks only care about time, since in the modern era it seems we have infinite space. The following is an excerpt from an email I recently sent:

> *I ran time*[6] *to see how much memory the algorithm takes. For n=10 and n=12, I get 0.65 GB and 9.33 GB, respectively. The algorithm is EXP in space, so I did a best fitting exponential. For n=14 (which is what is needed), the output is 132.5 GB. Yikes.*

[6] Unix time and memory command

Khovanov Homology

No working totally free-and-open-source implementation of a decent Khovanov homology algorithm is known to me. There are three existing versions, all open source, but each has a problem.

- One is written in Mathematica, which is not an open language.[7]
- Another is written in an old version of Java which wouldn't compile under the latest version of the language.[8]
- The sage implementation requires a super computer to actually use (see previous email).

All efforts are greatly appreciated, but there needs to be a *usable* implementation available to the general public without costing an arm and a leg.

[7] Current pricing options are \$19/month or \$183/year. Ouch.

[8] Edit: Nikolay Pultsin recently fixed this and it compiles with OpenJDK-17.

Khovanov Homology

Why might we care?

- Khovanov homology is an unknot detector! An efficient algorithm gives a decent means of solving the unknotting problem.
- It is a very strong invariant. When trying to tabulate a list of all knots up to N crossings, Khovanov homology can be used to help remove duplicates.
- Testing conjectures!

An Algorithm for the Jones Polynomial

We'll first start with smaller means. We'll tackle the Jones polynomial. We'll do this by computing the Kauffman bracket. Using the recursive definition we can inductively prove the following formula:

$$\langle L \rangle = \sum_{n=0}^{2^N-1} (-q)^{w(n)} (q + q^{-1})^{c(n)} \tag{11}$$

Here, $w(n)$ is the *Hamming weight* of n, the number of 1's that occur in the binary representation of n. $c(n)$ is the *circle counting function*, the number of disjoint circles that result from the complete resolution of L corresponding to n. To compute $\langle L \rangle$ we need only compute $c(n)$.

An Algorithm for the Jones Polynomial

First, thicken the knot. All of the crossings then become "four-way intersections."

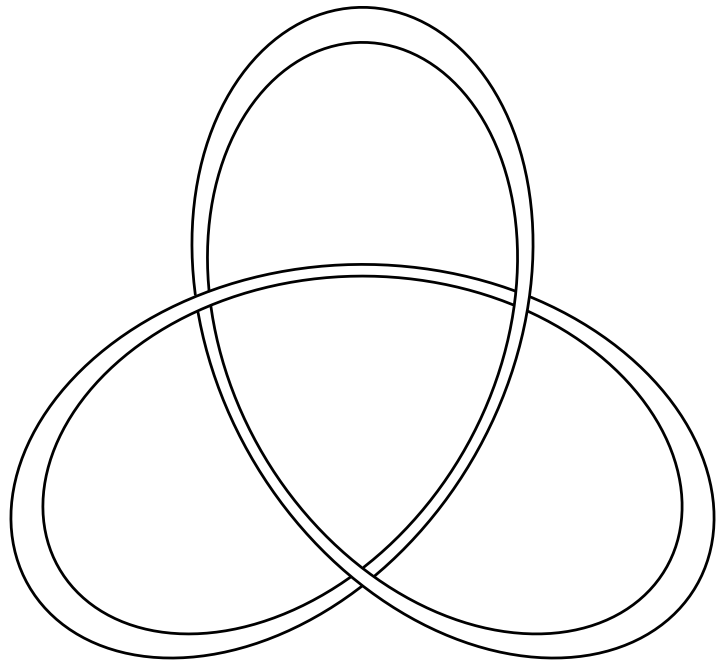

Figure: Framed Left-Handed Trefoil

An Algorithm for the Jones Polynomial

The aforementioned four-way intersections.

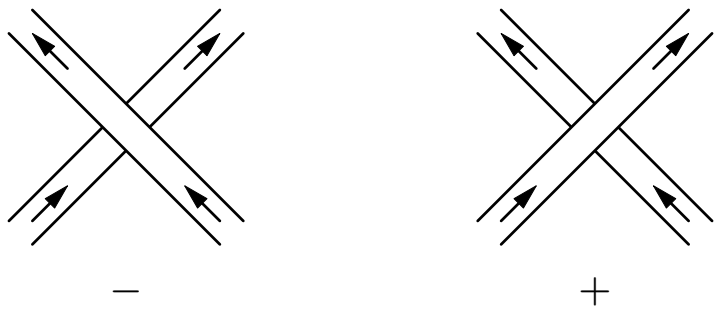

Figure: Signed Crossings in a Framed Knot

An Algorithm for the Jones Polynomial

A smoothing amounts to a roadblock.

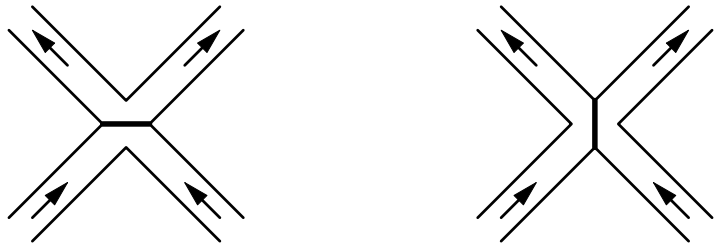

Figure: Smoothing a Negative Crossing in a Framed Knot

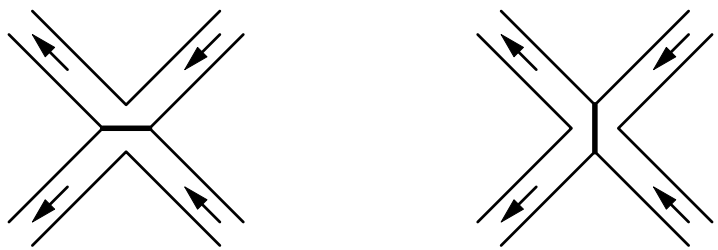

Figure: Smoothing a Positive Crossing in a Framed Knot

An Algorithm for the Jones Polynomial

Life will be easier if we label the four-way intersection. Given a positive or negative crossing, we will label the four roads as follows:

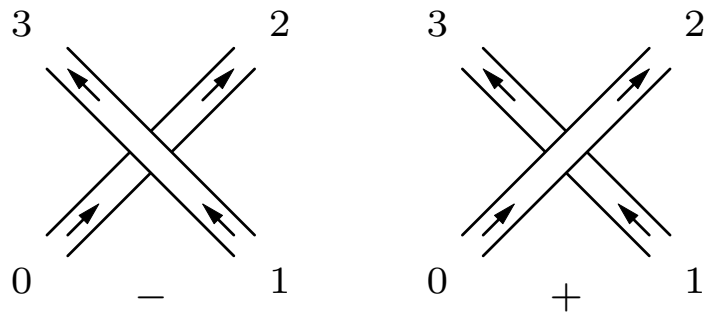

Figure: Thickened Crossings with Labels

Figure: Thickened Resolved Crossings with Labels

An Algorithm for the Jones Polynomial

Create a table with $4N$ entries, all of which are set to 0. Start at the 0 crossing in the Gauss code. Pictorially, you are walking towards the crossing from road 0. You now need to know which road to leave from. Let's suppose the sign of the crossing is negative, and you are supposed to do the zero resolution for this crossing. The previous figure shows that we must travel down road 1. But hold on, the arrows are pointing the wrong way! So we must walk backwards.

What does this mean? Find the next entry in the Gauss code for the 0 crossing (If the first entry is $O0-$, find $U0-$, and vice-versa). Since we are walking backwards, from there go to the *previous* entry in the Gauss code (that's what walking backwards means). We have traversed roads 0 and 1 for the 0 crossing, so change entries 0 and 1 of our table to 1.

An Algorithm for the Jones Polynomial

We continue this idea for all other crossings. We just need to know which road to leave from, given the crossing sign, type, and resolution, and which road we are entering from for the next crossing. This can be obtained by studying the previous figures carefully, but it is summarized in the following tables.

An Algorithm for the Jones Polynomial

In	Sign	Resolution	Out
0	-	0	1
0	-	1	3
0	+	0	3
0	+	1	1
1	-	0	0
1	-	1	2
1	+	0	2
1	+	1	0
2	-	0	3
2	-	1	1
2	+	0	1
2	+	1	3
3	-	0	2
3	-	1	0
3	+	0	0
3	+	1	2

Table: The Circle Counting Algorithm - Where to Go

An Algorithm for the Jones Polynomial

Type	Sign	Direction	In
O	-	Forward	1
O	-	Backward	3
O	$+$	Forward	0
O	$+$	Backward	2
U	-	Forward	0
U	-	Backward	2
U	$+$	Forward	1
U	$+$	Backward	3

Table: The Circle Counting Algorithm - Where to Start

An Algorithm for the Jones Polynomial

Eventually we will get back to road 0 of the zeroth crossing. Make sure to keep track of which roads you've travelled. If you've entered or exited through road $0 \leq k \leq 3$ of the n^{th} crossing, change the $4n + k$ entry of the table to one.

After you've completed your cycle, find the first entry in the table that is still zero and repeat this process. After at most $4N$ steps, you'll be done. The number of cycles you've counted is the number of circles that resulted from the given resolution.

An Algorithm for the Jones Polynomial

Let's use the trefoil as an example. The 000_2 resolution results in 2 circles (Fig. 11). Let's check that the algorithm detects this. The Gauss code is:

$$O0 + U1 + O2 + U0 + O1 + U2+ \tag{12}$$

We start by entering the zeroth crossing (road 0). It is an over crossing, so we look ahead in the code and find the corresponding under crossing. It is a positive crossing with the zero resolution, so Tab. 1 tells us to leave through road 3. We mark road 0 and road 3 of the zeroth crossing as travelled and proceed. Leaving through road 3 means we travel forward. Hence we wind up at $O1+$ in the code and we are walking forwards. Over-crossing, positive sign, walking forwards means we are entering the crossing from road 0 (Tab. 2).

An Algorithm for the Jones Polynomial

$$O0 + U1 + O2 + U0 + O1 + U2+ \tag{13}$$

We will again leave through road 3 and enter the $O2+$ crossing walking forwards. We again enter road 0 and leave through road 3 and wind up at $O0+$ walking forward, completing our first cycle. The next untouched road is road 1 for the zeroth crossing. This corresponding to $U0+$ walking forwards. Tab. 1 tells us to leave through road 2. We end up at $U1+$ at road 1. Again we leave through road 2 and end up at $U2+$ road 1. We leave through road 3 entering $U0+$, completing our cycle. All of the roads have been marked, and we have 2 circles total, in agreement with Fig. 11.

An Algorithm for the Jones Polynomial

The simplest knot to try this on in it's entirety by hand is the chain-link-fence knot. You must use the circle counting algorithm 4 times, one for each resolution. Try it!

An Algorithm for the Jones Polynomial

Benefits:

- Time complexity is $O(N2^N)$, so no worse than the recursive algorithm.
- The algorithm is $O(N)$ in space! Significantly better than $O(2^N)$. The reason being we don't need to store all resolutions in memory. We can do one at a time, add the result to our polynomial, and continue.
- The idea should generalize to Khovanov homology. We need only count the circles.
- The idea works for virtual knots. There is no restriction to the classical setting.

Conjectures on Khovanov and Knot Floer Homology

A Legendrian knot is an embedding of $\mathbb{S}^1$ into $\mathbb{R}^3$ that is everywhere tangent to the standard contact structure.

Every knot is topologically equivalent to a Legendrian knot. Two Legendrian knots are said to be equivalent if they are equivalent through a homotopy of Legendrian knots.

It is possible for inequivalent Legendrian knots to be equivalent as topological knots.

If Legendrian knots in a given topological knot type are uniquely determined by two classical invariants (their Thurston-Bennequin numbers and rotation numbers), they are said to be Legendrian simple.

Conjectures on Khovanov and Knot Floer Homology

It is known that torus knots are Legendrian simple.

Two results have shown that Khovanov homology (Mrowka and Kronheimer) and Knot Floer homology (Ozváth and Szabó) are unknot detectors. The unknot being the simplest of the torus knots, a somewhat natural generalization would be that these two homology theories distinguish the Legendrian simple knots.

Using the algorithm outlined, numerical evidence for this has been collected for all knots up to 17 crossings (roughly 8 million knots).[9]

[9] **Correction:** There was a bug in the code. The conjecture fails for KFH.

Conjectures on Khovanov and Knot Floer Homology

An open analogous question for the Jones polynomial is whether it too detects the unknot. This is not true of torus knots. There are non-torus knots with the same Jones polynomial as a torus knot.

- $T(2,5)$ matches a 10 crossing knot.
- $T(2,7)$ matches a 12 crossing knot.
- $T(2,11)$ matches a 14 crossing knot.

Any patterns? $T(n,m)$ is always of the form $(2, \text{prime})$, the second number is always increasing by 2. Hmm.

Conjectures on Khovanov and Knot Floer Homology

- $T(2,5)$ matches a 17 crossing knot.

Well dang, there's goes that trend.

Still, $T(n,m)$ is always of the form $(2, \text{prime})$. It would be nice to know if there are infinitely many of these matches.

For the 10, 12, and 14 crossing matches, the Khovanov homologies are different.

The 17 crossing knot can not be computed yet since no available computer to me has the required 500+ GB of memory needed.

Hence the need for an iterative implementation that is $O(N)$ in space!

Conjectures on Khovanov and Knot Floer Homology

A similar search through the all knots up to 12 crossings yielded no match for Knot Floer homology.[10]

Much the way Khovanov homology is the categorification of the Jones polynomial, so is Knot Floer homology for the Alexander polynomial.

The Alexander polynomial is first computed (this is polynomial time, much better than the Jones polynomial), and if matches were found the Knot floer homologies were compared as well (slow, exponential in time).

[10] **Correction:** Matches were found once the code was fixed.

References


Knot atlas dt codes.
`http://katlas.math.toronto.edu/wiki/DT_%28Dowker-Thistlethwaite%29_Codes`.
Accessed: 2023-02-27.


C.H. Dowker and Morwen B. Thistlethwaite.
Classification of knot projections.
Topology and its Applications, 16(1):19–31, 1983.