# VIR Results

Duncan Wilkie et. al.

1 April 2023

The project concerns the Floer homology of 1-1 knots. Knots of this type can be drawn on a torus, and represented via a 4-tuple $[p, a, b, r]$, from which the knot can be drawn on the torus (as a quotient of $\mathbb{R}^2$) in a Heegaard diagram. Computing the homology from this diagram is very easy—precisely the reason for the study of this topic.

## 1 Drawing

Code is being written which takes as input the 4-tuple and draws the Heegaard diagram in TikZ. The correctness of this code depends firstly on the correctness of the underlying tooling, which we shall not verify, but also on several semantic propositions about relationships between sides of the diagrams.

Those we may formulate mathematically, and attempt to prove.

The code concerns in large part the relationship between the generators and a coordinate system in TikZ. Given a Heegaard diagram drawn on a square, this coordinate system is imposed as follows: the vertical or $y$ axis lies along the left side of the diagram, and the horizontal or $x$ axis lies along the very bottom of the diagram. The scale for the axes is such that the top of the square diagram is the line $y = p + 1$, and the point corresponding to the $k$th generator on the left side lies at $y = p + 1 - k$.

The drawing of a single diagram is done via `\diagram{p}{a}{b}{r}`; in the macro body, these arguments become `#1, #2, #3, #4`. Most of the logic is done using the TikZ-math extension, which permits basic integer and modular integer arithmetic.

There is initial setup of some macro constants when the function is called:

```
\points = #1;
\maxy = \points + 1;
\maxx = \points;
\discs = #2;
\abov = #3;
\belo = \points - 2*\discs - \abov;
\shift = #4 - 1;
\discmidl = \maxy - \discs - 0.5; % y-coordinate of midpoint of left-side rainbow arcs
\discmidr = \maxy - mod(\abov + \discs + \shift, \points) - 0.5;
% here-down: stuff related to the warping arcs
\innerdisctop = \abov + 1 + \shift; % un-wrapped n-value of the top (i.e. wrt. y-coordinate)
                                    % of the innermost disc on the right
```

```
if \innerdisctop > \points then { % there exist fully warped discs
  \warpdiscs = \discmidr - 0.5; % number of fully warped discs (need to check)
} else {
  \warpdiscs = 0;
};
\innerdiscbot = mod(\innerdisctop + 2 * \discs - 1, \points) + 1;
\innerdisctop = mod(\innerdisctop - 1, \points) + 1;
% there are precisely \shift - 2 * \warpdiscs warping arcs
% use this to calculate parameters needed to make the warps line up
\warpcnt = \shift - 2 * \warpdiscs;
if \warpcnt == 1 then {
  \warpstart = \maxx / 2;
  \warpend = \maxy / 2;
  \warpstep = 0;
} else {
  \warpstep = (\maxx / 2) / (\warpcnt - 1);
  \warpstart = \maxx / 4;
  \warpend = 3 * \warpstart;
};
```

Then, we define some convenience functions to avoid shortcomings of the DSL:

```
% produce a coordinate given a pair and an axis
% (necessary because coordinates cannot be returned from functions)
function proj(\vala, \valb, \whaxis) {
  if \whaxis == 0 then {
    return \vala;
  } else {
    return \valb;
  };
};
% We want to work as abstractly as possible, namely,
% with the Heegaard generator numberings.
% This function converts those generator numberings, along with which side one is on,
% to TikZ coordinates.
function ntoxy(\n, \side, \axis) {
  if \side == 0 then {
    return proj(0, \maxy - \n, \axis);
  } else {
    return proj(\maxx, \maxy - \n, \axis);
  };
};
```

Now, we can get into some content. The following produces the right-hand points corresponding to a left-hand point before applying the `\shift`; this helps by not needing to differentiate between above and below arcs in the drawing logic.

```
% given a point on the left side, return the endpoint number
% of the corresponding arc on the right side in the unshifted diagram.
```

```
function imagenoshift(\n) {
  if \n <= 2 *\discs then { % disc point
    return \abov + \n;
  } else {
    if \n <= \points - \belo then { % above arc
      return \n - 2 * \discs;
    } else { % below arc
      return \n;
    };
  };
};
```

Now, for the actual drawing. This function draws everything from the left side:

```
function drawleft(\n) {
  \sx = ntoxy(\n, 0, 0);
  \sy = ntoxy(\n, 0, 1);
  if \n <= 2 * \discs then { % disc point; no warp possible
    \ex = abs(\n - \discs - 0.5);
    \ey = \discmidl;
    {
      \node[dot] (c) at (\sx,\sy) {};
      \draw (\sx,\sy) -- (\ex,\ey);
    };
  } else { % arc goes to right side; warp possible
    \otheren = imagenoshift(\n) + \shift;
    if \otheren > \points then { % warps
      \ex = \warpstart + (\points - \n) * \warpstep;
      \otherny = ntoxy(mod(\otheren - 1, \points) + 1, 1, 1);
      {
        \node[dot] (c) at (\sx,\sy) {};
        \draw (\sx, \sy) -- (\ex,0);
        \draw (\ex,\maxy) -- (\maxx,\otherny);
      };
    } else {
      \otherny = ntoxy(mod(\otheren - 1, \points) + 1, 1, 1);
      {
        \node[dot] (c) at (\sx,\sy) {};
        \draw (\sx,\sy) -- (\maxx,\otherny);
      };
    };
  };
};
```

And this one takes care of what that misses:

```
function drawright(\n) {
  \preimage = mod(\n - \shift - 1, \points) + 1;
```

```
    \preerimage = mod(\preimage - \above - 1, \points) + 1;
    \sx = ntoxy(\n, 1, 0);
    \sy = ntoxy(\n, 1, 1);
    if \preerimage <= 2 * \discs then { % disc point
      \delta = 2 * (abs(\preerimage - \discmidl) - 0.5) + 1;
      if \preerimage <= \discs then { % upper half
        \otherend = \n + \delta;
        if \otherend > \points then { % warps
          \ex = \warpstart + ()
          {
            \draw (\sx,\sy) --
          };
        } else { % doesn't warp
          \ex = abs(\maxx - (\n - \discs - 0.5));
          \ey = \discmidr;
          {
            \draw (\sx,\sy) -- (\ex,\ey);
          };
        };
      } else { % lower half
        \otherend = \n - \delta;
        if \otherend < 1 then { % warps
          {
p            \draw
          };
        } else { % doesn't warp
          \ex = abs(\maxx - (\n - \discs - 0.5));
          \ey = \discmidr;
          {
            \draw (\sx,\sy) -- (\ex,\ey);
          };
        };
      };
    };
```

With this set up, it's a simple matter of iterating over the points, calling these functions, and adding details.

```
    % now actually draw it
    for \i in {1,...,\points}{
      drawleft(\i);
      % drawright(\i);
    };
    {
      % bounding square
      \draw (0,0) rectangle (\maxx,\maxy);
```

```
    };
```

The actual user-facing code embellishes and combines these diagrams to produce high-quality substrates for contentual mathematical reasoning: the user is expected to use

```
% Put the diagram in a tikzpicture and draw the node labels
\newcommand{\heegaard}[4]{
  \begin{tikzpicture}[dot/.style = {circle, fill, minimum size=6pt, inner sep=0pt, outer sep=0pt}, c
    \diagram{#1}{#2}{#3}{#4}
    \draw (0,0) node foreach \i in {1,...,#1} at (-0.5,#1-\i+1) {$\i$};
    \draw (0,0) node foreach \i in {1,...,#1} at (#1+0.5,#1-\i+1) {$\i$};
  \end{tikzpicture}
}
```

for standalone diagrams (which merely adds generator numberings) and

```
% % Given [p, a, b, r] and an x and a y, make an x-by-y grid of Heegaard diagrams.
\newcommand{\ucover}[6]{
  \tikzset{
    tile/.pic={
      \diagram{#1}{#2}{#3}{#4}
    }
  }
  \begin{tikzpicture}[dot/.style = {circle, fill, minimum size=6pt, inner sep=0pt, outer sep=0pt}, c
    \foreach \x in {1,...,#5}  {
      \foreach \y in {1,...,#6} {
        \tikzmath{
          \xorig = (\x - 1) * #1;
          \yorig = (\y - 1) * (#1 + 1);
        }
        \pic (S) at (\xorig,\yorig) {tile};
      };
    };
  \end{tikzpicture}
}
```

for drawing rectangular subsets of the universal cover diagram.

**Proposition 1.** *The following tikzmath function converts a generator number, a side number, and an axis number to the corresponding coordinate of the generator in the above-described system. The side number is either 0, corresponding to a left-side generator, or 1, corresponding to a right-side generator. The axis number is either 0, corresponding to the horizontal coordinate, or 1, corresponding to the vertical coordinate.*

**Proposition 2.** *The following tikzmath function converts a generator number on the left side to*

**Proposition 3.** *The*