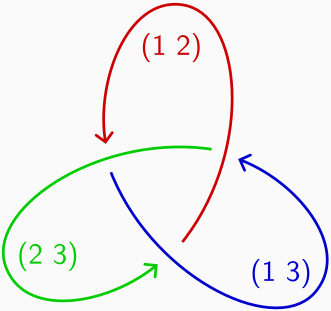
Group Colourings of Knots

Blake Freer

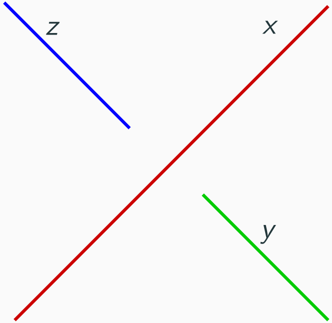
December 3, 2024



The trefoil coloured with S_3

Let p be prime. A **mod-p colouring** of a knot is a labelling of the arcs with the integers $0, \ldots, p-1$ such that

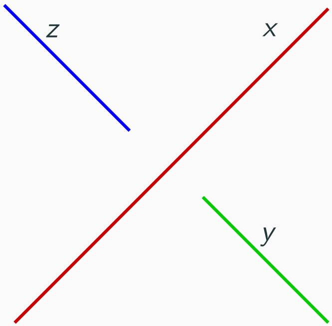
$$2x - y - z \equiv 0 \mod p$$



p-colourings

Let p be prime. A **mod-p colouring** of a knot is a labelling of the arcs with the integers $0, \ldots, p-1$ such that

$$2x - y - z \equiv 0 \mod p$$



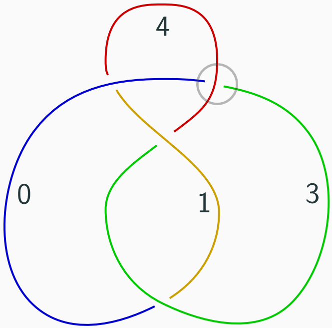
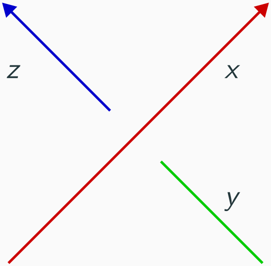


Figure 1: A 5-colouring of 4_1

Wirtinger Presentation

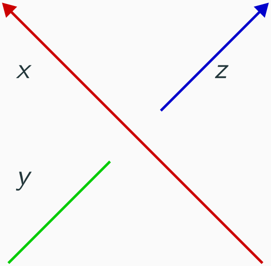
The Wirtinger presentation of $\pi_1(K)$ has the arcs as generators and crossings as relations.

$$\pi_1(K) = \langle x_1, x_2, \ldots x_n \mid r_1, r_2, \ldots, r_n \rangle$$



Positive Crossing

$z = x^{-1}yx$



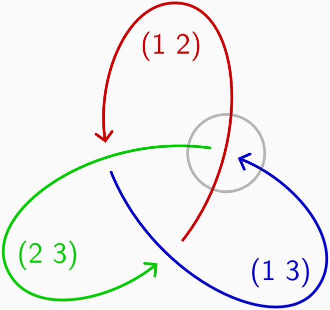
Negative Crossing

$z = xyx^{-1}$

Group Colourings

Definition

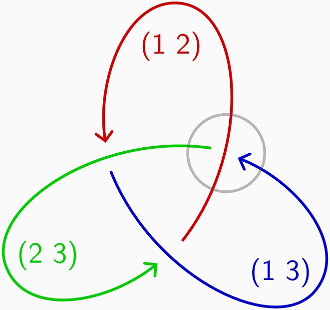
Let G be a group. A G-**colouring** of a knot is a labelling of the arcs with elements of G which satisfies the Wirtinger relation at each crossing.



Group Colourings

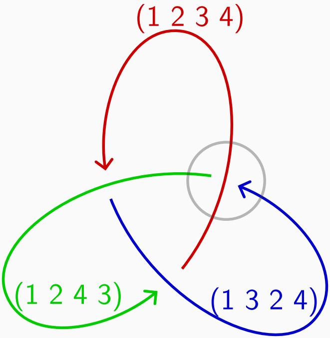
Definition

Let G be a group. A G-**colouring** of a knot is a labelling of the arcs with elements of G which satisfies the Wirtinger relation at each crossing.



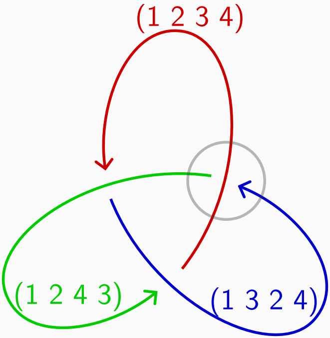
$$
\begin{aligned}
&(1\ 2)^{-1}(1\ 3)(1\ 2)\\
=&(2\ 1)(1\ 3)(1\ 2)\\
=&(1\ 2\ 3)(1\ 2)\\
=&(2\ 3)
\end{aligned}
$$

S_4-colouring



$$
\begin{aligned}
&(1\ 2\ 3\ 4)^{-1}(1\ 3\ 2\ 4)(1\ 2\ 3\ 4)\\
=&(4\ 3\ 2\ 1)(1\ 3\ 2\ 4)(1\ 2\ 3\ 4)\\
=&(2\ 3\ 4)(1\ 2\ 3\ 4)\\
=&(1\ 2\ 4\ 3)
\end{aligned}
$$

S_4-colouring



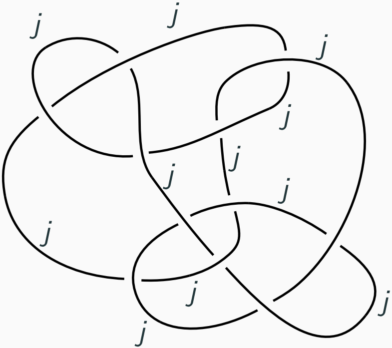
$$
\begin{aligned}
&(1\ 2\ 3\ 4)^{-1}(1\ 3\ 2\ 4)(1\ 2\ 3\ 4)\\
=&(4\ 3\ 2\ 1)(1\ 3\ 2\ 4)(1\ 2\ 3\ 4)\\
=&(2\ 3\ 4)(1\ 2\ 3\ 4)\\
=&(1\ 2\ 4\ 3)
\end{aligned}
$$

Theorem (Perko 1975)

A knot is S_3-colourable if and only if it is S_4-colourable.

Trivial Colourings

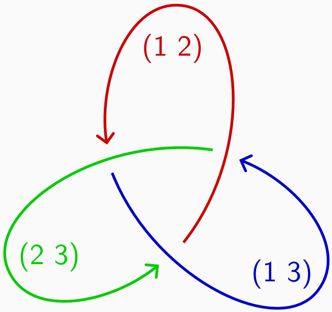
Labelling every arc with the same $g \in G$ trivially satisfies the Wirtinger relations. We typically ignore these.



We would not say that the Conway knot is Q_8-colourable.

Trivial Colourings

We require that the labels generate G, i.e. the labelling is a *surjective homomorphism* of $\pi_1(K)$ onto G.



This is not an S_4 colouring, even though all of the labels are in S_4.

Dihedral Colouring

The **dihedral group** D_n is the symmetries of the regular n-sided polygon. It has order $|D_n| = 2n$ and a finite presentation

$$D_n = \left\langle r, s \mid s^2 = r^n = 1, sr^k = r^{-k}s \right\rangle$$

The **dihedral group** D_n is the symmetries of the regular n-sided polygon. It has order $|D_n| = 2n$ and a finite presentation

$$D_n = \left\langle r, s \mid s^2 = r^n = 1, sr^k = r^{-k}s \right\rangle$$

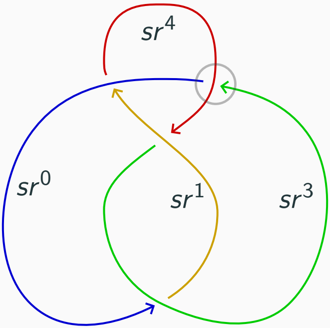


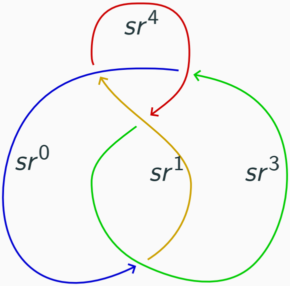
Figure 2: D_5-colouring of 4_1.

$$\begin{aligned} & sr^4 sr^3 (sr^4)^{-1} \\ =& sr^4 r^{-3} s r^{-4} s \\ =& sr^{4-3} r^4 s s \\ =& sr^{4+4-3} \\ =& sr^5 \\ =& sr^0 \end{aligned}$$

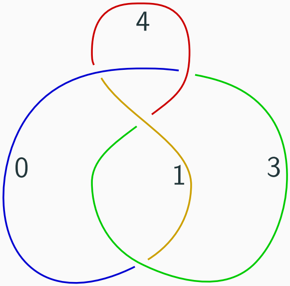
Fox-n Colourings

Fox-n Colourings

D_5-colouring of 4_1



mod-5 colouring of 4_1

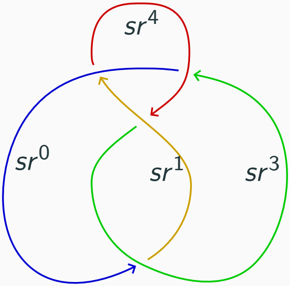


D_5-colouring of 4_1 mod-5 colouring of 4_1

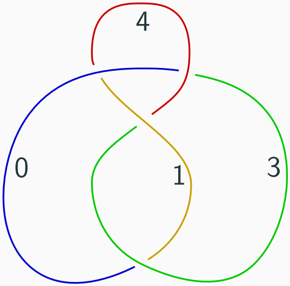
The mod-p colourings are dihedral group colourings in disguise!

Fox-n Colourings

D_5-colouring of 4_1



mod-5 colouring of 4_1

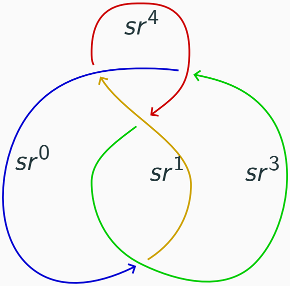


The mod-p colourings are dihedral group colourings in disguise!

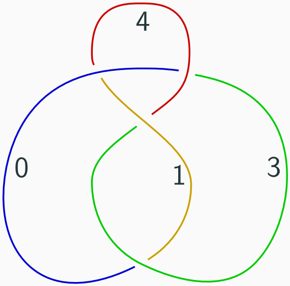
The function $i \mapsto sr^i$ takes a mod-p label to a D_p label.

Fox-n Colourings

D_5-colouring of 4_1



mod-5 colouring of 4_1



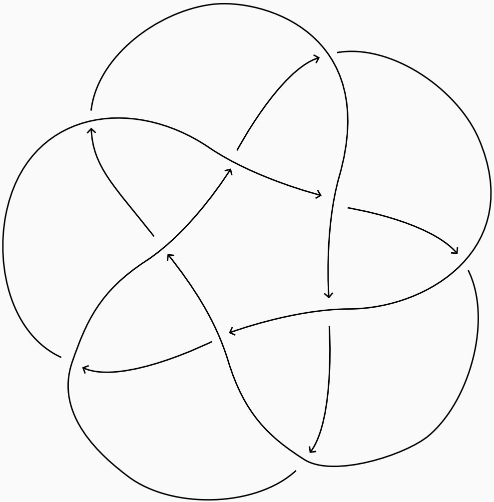
The mod-p colourings are dihedral group colourings in disguise!

The function $i \mapsto sr^i$ takes a mod-p label to a D_p label.

"$\equiv 0 \mod p$" is replaced by $r^p = 1$.

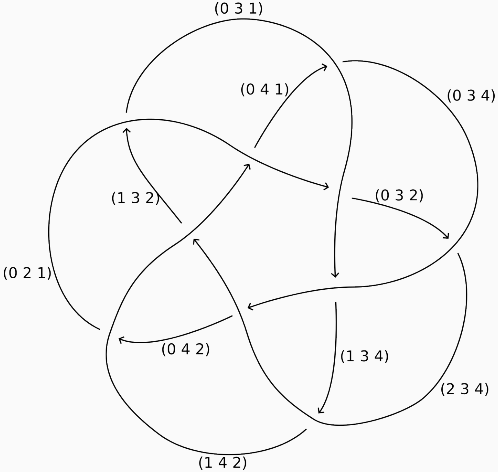
Application: $T_{5,3}$

The $(5, 3)$ torus knot has determinant $\det(T_{5,3}) = 1$, so it is not mod-p colourable for *any* prime p.



Application: $T_{5,3}$

$T_{5,3}$ has a non-trivial colouring with A_5, the alternating permutations on 5 letters.



Application: $T_{5,3}$

```python
# Blake Freer, 2024
# https://github.com/BlakeFreer

from itertools import product

from sympy.combinatorics import Permutation, PermutationGroup
from sympy.combinatorics.named_groups import AlternatingGroup

GROUP = AlternatingGroup(5)


def try_solve(x0, x4, x8):
    x1 = (~x4) * x0 * x4
    x5 = (~x8) * x4 * x8
    x2 = (~x8) * x1 * x8
    x9 = (~x2) * x8 * x2
    x6 = (~x2) * x5 * x2
    x3 = (~x6) * x2 * x6
    x7 = (~x0) * x6 * x0

    assert x0 == (~x6) * x9 * x6, "Inconsistent at x0"
    assert x4 == (~x0) * x3 * x0, "Inconsistent at x4"
    assert x8 == (~x4) * x7 * x4, "Inconsistent at x8"

    solution = [x0, x1, x2, x3, x4, x5, x6, x7, x8, x9]

    assert PermutationGroup(*solution).equals(GROUP), "Doesn't generate."

    return solution
```

Application: $T_{5,3}$

```python
def solve_conj_class(conj: set[Permutation]):
    solutions = []
    for x0, x4, x8 in product(conj, repeat=3):
        try:
            solutions.append(try_solve(x0, x4, x8))
        except AssertionError:
            pass
    return solutions


for cls in GROUP.conjugacy_classes():
    solutions = solve_conj_class(cls)
    print(
        f"Found {len(solutions)} colourings with the conjugacy class of {list(cls)[0]}."
    )

    if solutions:
        print("One example is:")
        for idx, p in enumerate(solutions[0]):
            print(f"x{idx} = {p}")
```