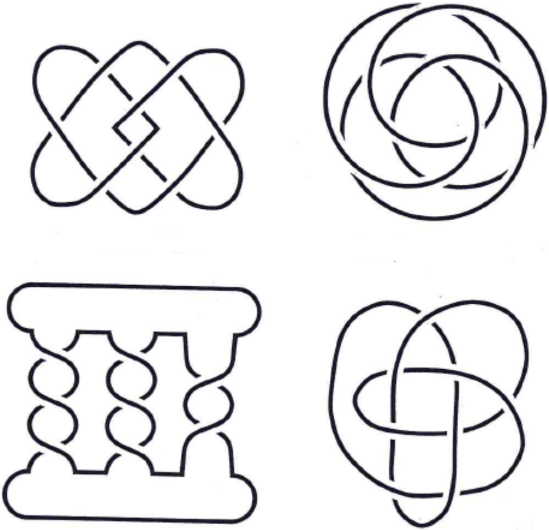
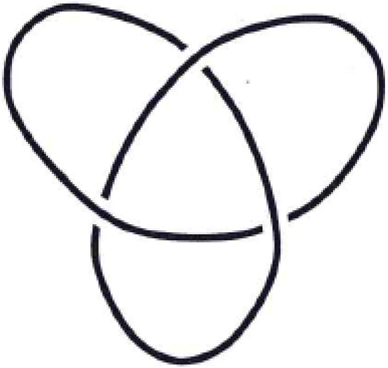
Is it really knotted?

Four pictures, one knot



Is it really knotted?



If you think it cannot be untangled, PROVE IT!

Knot recognition

Knot equivalence = a continuous deformation of the space that transforms one knot into the other.

Fundamental Problem

Given two knots (or knot diagrams), are they equivalent?

Is it (algorithmically) decidable?

If so, what is the complexity?

Knot recognition

Knot equivalence = a continuous deformation of the space that transforms one knot into the other.

Fundamental Problem

Given two knots (or knot diagrams), are they equivalent?

Is it (algorithmically) decidable?

Yes, very hard to prove. (Haken, 1962)

If so, what is the complexity?

Nobody knows. No provably efficient algorithm known.

Knottedness

Knot equivalence = a continuous deformation of space that transforms one knot into the other.

Fundamental Problem

Given a knot (or a knot diagram), is it equivalent to the plain circle?

Is it (algorithmically) decidable?

If so, what is the complexity?

Knottedness

Knot equivalence = a continuous deformation of space that transforms one knot into the other.

Fundamental Problem

Given a knot (or a knot diagram), is it equivalent to the plain circle?

Is it (algorithmically) decidable?

Yes, hard to prove. (Haken, < 1962)

If so, what is the complexity?

Nobody knows. No provably efficient algorithm known.
Known to be in NP $\cap$ coNP (under GRH).

[Hass-Lagarias-Pippenger 1999, Lackenby 2015; Kuperberg 2014]

Complexity classes P, NP, coNP

Consider a decision problem (e.g., knot equivalence, or primeness).

P = there is a polynomial-time algorithm that decides the problem for every input

NP = for every input with *positive* answer, there is a certificate that can be verified in polynomial time

coNP = for every input with *negative* answer, there is a certificate that can be verified in polynomial time

Complexity classes P, NP, coNP

Consider a decision problem (e.g., knot equivalence, or primeness).

P = there is a polynomial-time algorithm that decides the problem for every input

NP = for every input with *positive* answer, there is a certificate that can be verified in polynomial time

coNP = for every input with *negative* answer, there is a certificate that can be verified in polynomial time

Example: problem: is a given number n prime?

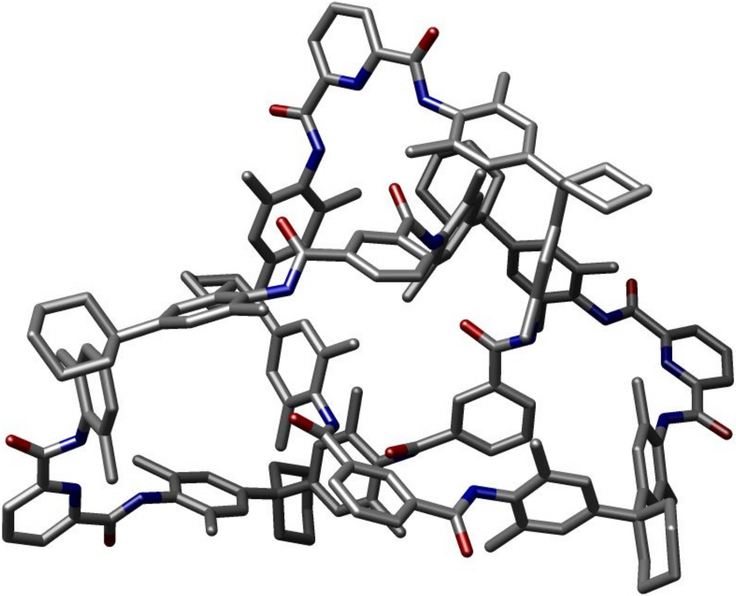
- coNP: m such that $1 \neq m \mid n$
- NP: m that is coprime to n and $ord(m) = n - 1$ in $\mathbb{Z}_n^*$
- P: a complicated algorithm from 2002

What is knot recognition good for?

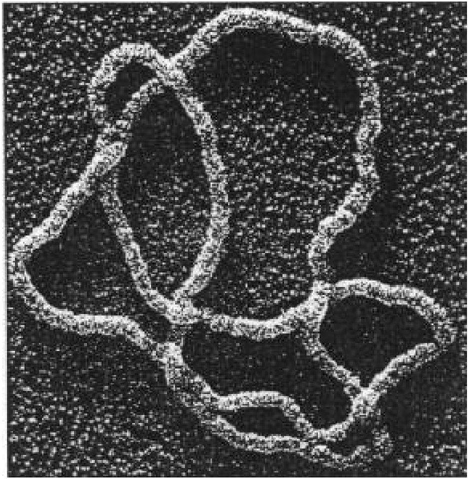
What is knot recognition good for?

(I don't care too much.)

Knots are in chemistry



Knots are in biology



... with applications towards antibiotics production (believe or not)

Knots are everywhere

... with applications towards black magic (believe or not)

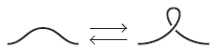
Reidemester moves

Knots are usually displayed by a *regular* projection into a plane.

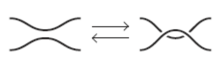
Theorem (Reidemeister 1926, Alexander-Brigs 1927)

$K_1 \sim K_2$ if and only if they are related by a finite sequence of Reidemeister moves:

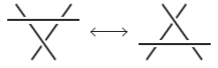
I. *twist/untwist a loop;*



II. *move a string over/under another;*

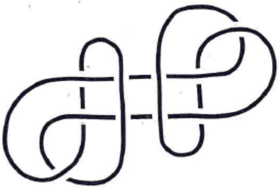


III. *move a string over/under a crossing.*



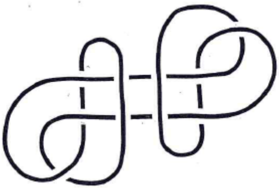
Reidemeister moves, where is the problem?

Bad news: When unknotting, $\mathrm{cross}(K)$ may increase



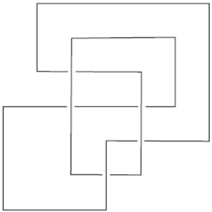
Reidemeister moves, where is the problem?

Bad news: When unknotting, $\mathrm{cross}(K)$ may increase



Good news: Lackenby (2015): not too much... $\leq 49 \cdot \mathrm{cross}(K)^2$

Lackenby's idea: a special type of diagrams and moves (Dynnikov's theory)



Reidemeister moves, algorithmically?

Fact

Assume there is a computable function $f(n)$ that bounds the number of Reidemeister moves to transform equivalent diagrams with $\leq n$ x-ings. Then knot equivalence is decidable.

Finding such $f(n)$ is very difficult:

- Coward-Lackenby (2014): $\exists f$ computable (extremely fast growing)

Special case $K_2 = \bigcirc$:

- Hass-Lagarias (2001): f exponential, $f(n) = 2^{10^{11} n}$
- Lackenby (2015): f polynomial, $f(n) = (236n)^{11}$

Reidemeister moves, algorithmically?

Fact

Assume there is a computable function $f(n)$ that bounds the number of Reidemeister moves to transform equivalent diagrams with $\leq n$ x-ings. Then knot equivalence is decidable.

Finding such $f(n)$ is very difficult:

- Coward-Lackenby (2014): $\exists f$ computable (extremely fast growing)

Special case $K_2 = \bigcirc$:

- Hass-Lagarias (2001): f exponential, $f(n) = 2^{10^{11} n}$
- Lackenby (2015): f polynomial, $f(n) = (236n)^{11}$

- Hass-Nowik (2010): quadratic lower bound for unknot diagrams … $\exists K^{(n)} \sim \bigcirc$, $n = \mathrm{cross}(K^{(n)})$, with at least $n^2/25$ moves

Recognizing knots, summary

Fundamental Problem

Given K_1, K_2, are they equivalent?

- Haken (1961): $\sim \bigcirc$ is decidable (in EXP-time)
- Haken (1962): $\sim$ is decidable (in EXP-time)
- Hass-Lagarias-Pippenger (1999): $\sim \bigcirc$ is in NP (certificate: certain normal surface)
- Coward-Lackenby (2014): $\sim$ is decidable by bounding Reidemeister moves
- Lackenby (2015): $\sim \bigcirc$ is in NP by bounding Reidemeister moves (certificate: a sequence of Reidemeister moves)

Recognizing knots, summary

Fundamental Problem

Given K_1, K_2, are they equivalent?

- Haken (1961): $\sim \bigcirc$ is decidable (in EXP-time)
- Haken (1962): $\sim$ is decidable (in EXP-time)
- Hass-Lagarias-Pippenger (1999): $\sim \bigcirc$ is in NP (certificate: certain normal surface)
- Coward-Lackenby (2014): $\sim$ is decidable by bounding Reidemeister moves
- Lackenby (2015): $\sim \bigcirc$ is in NP by bounding Reidemeister moves (certificate: a sequence of Reidemeister moves)

- Agol (2002, not published): $\sim \bigcirc$ is in coNP assuming GRH
- Kuperberg (2014): $\sim \bigcirc$ is in coNP assuming GRH

Proving impossibility (i.e., certifying inequivalence)

Problem: Given $K_1 \not\sim K_2$, prove it!

... example:  $\not\sim \bigcirc$!

... develop *invariants*, properties shared by equivalent knots:

$$K_1 \sim K_2 \quad \text{implies} \quad P(K_1) = P(K_2)$$

... if $P(K_1) \neq P(K_2)$, then P is a certificate of inequivalence

Proving impossibility (i.e., certifying inequivalence)

Problem: Given $K_1 \not\sim K_2$, prove it!

... example:  $\not\sim$ ○ !

... develop *invariants*, properties shared by equivalent knots:

$$K_1 \sim K_2 \quad \text{implies} \quad P(K_1) = P(K_2)$$

... if $P(K_1) \neq P(K_2)$, then P is a certificate of inequivalence

Classical invariants use various algebraic constructions to code some of the topological properties of a knot.

- the fundamental group of the knot complement
- the Alexander, Jones and other polynomials
- Heegaard-Floer homology, Khovanov homology, ...
- etc. etc. etc.

Trade-off between computational complexity and ability to recognize knots.

Alexander polynomial