Computational Application of a Transfer Algorithm to the Borromean Rings

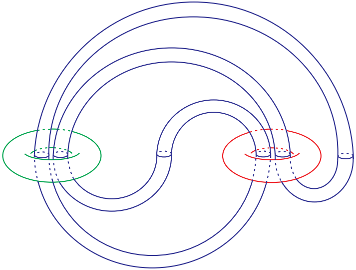
Merv Fansler

Millersville University of Pennsylvania

29 June 2016

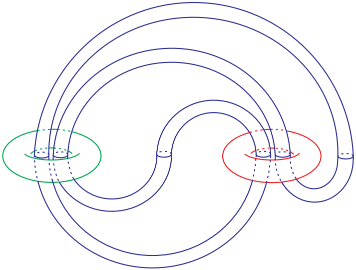
Advised by Ron Umble

Background



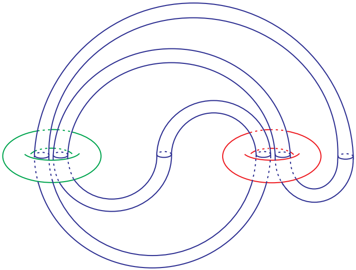
- started as summer (2015) project proposed by Ron Umble

Background



- started as summer (2015) project proposed by Ron Umble
- algorithm to possibly detect linkage in Brunnian links

Background



- started as summer (2015) project proposed by Ron Umble
- algorithm to possibly detect linkage in Brunnian links
- requires some algebraic topology and probably a lot of computation

Table of Contents

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)
  - closed balls (solids or 3-cells)

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)
  - closed balls (solids or 3-cells)
- glued together so that the

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)
  - closed balls (solids or 3-cells)
- glued together so that the
  - non-empty boundary of a k-cell is a union of $(k-1)$-cells

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)
  - closed balls (solids or 3-cells)
- glued together so that the
  - non-empty boundary of a k-cell is a union of $(k-1)$-cells
  - non-empty intersection of cells is a cell

Cellular Decomposition

- Let X denote a connected network, surface, solid or union thereof embedded in $\mathbb{R}^3$ or S^3
- A **cellular decomposition** of X is a finite collection of
  - discrete points (vertices or 0-cells)
  - closed intervals (edges or 1-cells)
  - closed disks (faces or 2-cells)
  - closed balls (solids or 3-cells)
- glued together so that the
  - non-empty boundary of a k-cell is a union of $(k-1)$-cells
  - non-empty intersection of cells is a cell
  - union of all cells is X

Chain Complex

A **chain complex** is

- a vector space $C(X)$ with basis $\{\text{cells of } X\}$, and

Chain Complex

A **chain complex** is

- a vector space $C(X)$ with basis $\{\text{cells of } X\}$, and
- a boundary operator $\partial : C(X) \to C(X)$ that is

Chain Complex

A **chain complex** is

- a vector space $C(X)$ with basis $\{\text{cells of } X\}$, and
- a boundary operator $\partial : C(X) \to C(X)$ that is
  - zero on vertices

Chain Complex

A **chain complex** is

- a vector space $C(X)$ with basis $\{\text{cells of } X\}$, and
- a boundary operator $\partial : C(X) \to C(X)$ that is
  - zero on vertices
  - linear on chains

Chain Complex

A **chain complex** is

- a vector space $C(X)$ with basis $\{\text{cells of } X\}$, and
- a boundary operator $\partial : C(X) \to C(X)$ that is
  - zero on vertices
  - linear on chains
  - a derivation of Cartesian product

Homology

- $H_*(C) = \mathrm{ker}\partial / \mathrm{Im}\partial$

Homology

- $H_*(C) = \ker\partial / \mathrm{Im}\partial$
- Elements of $H_*(C)$ are cosets $[c] = c + \mathrm{Im}\partial$

Homology

- $H_*(C) = \mathrm{ker}\partial / \mathrm{Im}\partial$
- Elements of $H_*(C)$ are cosets $[c] = c + \mathrm{Im}\partial$
- equivalence classes of nonbounding cycles that differ only by a boundary

Homology

- $H_*(C) = \ker\partial / \mathrm{Im}\partial$
- Elements of $H_*(C)$ are cosets $[c] = c + \mathrm{Im}\partial$
- equivalence classes of nonbounding cycles that differ only by a boundary
- Note: Homology alone **does not detect linkage**!

Diagonal Approximation

A map $\Delta : X \to X \times X$ is a **diagonal approximation** if

1. Δ is homotopic to Δ^G

Diagonal Approximation

A map $\Delta : X \to X \times X$ is a **diagonal approximation** if

1. Δ is homotopic to Δ^G
2. $\Delta(c)$ is a subcomplex of $c \times c$

Diagonal Approximation

A map $\Delta : X \to X \times X$ is a **diagonal approximation** if

1. Δ is homotopic to Δ^G
2. $\Delta(c)$ is a subcomplex of $c \times c$
3. ∂ is a coderivation of Δ, i.e., $\Delta\partial = (\partial \times \mathrm{Id} + \mathrm{Id} \times \partial)\Delta$

Coproduct Notation

Some notation:

- We will denote the diagonal approximation on chains by Δ_2

Coproduct Notation

Some notation:

- We will denote the diagonal approximation on chains by Δ_2
- Higher coproducts on chains will be denoted by subscripts, e.g., $\Delta_3, \Delta_4, \ldots$

Coproduct Notation

Some notation:

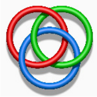
- We will denote the diagonal approximation on chains by Δ_2
- Higher coproducts on chains will be denoted by subscripts, e.g., $\Delta_3, \Delta_4, \ldots$
- Coproducts transferred to homology will be denoted by superscripts, e.g., $\Delta^2, \Delta^3, \ldots$

Brunnian Links

- A Brunnian link is a nontrivial link such that the removal of any component results in an unlink.

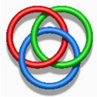
Brunnian Links

- A Brunnian link is a nontrivial link such that the removal of any component results in an unlink.
- Example: Borromean rings (3-component Brunnian link)



Brunnian Links

- A Brunnian link is a nontrivial link such that the removal of any component results in an unlink.
- Example: Borromean rings (3-component Brunnian link)



- We will denote the link complement in S^3 of an n-component Brunnian link by BR_n, $n \geq 3$.

Conjecture

Conjecture

A diagonal approximation Δ_2 on $C(BR_n)$ induces

- *a primitive diagonal $\Delta^2 : H(BR_n) \otimes H(BR_n)$,*
- *trivial k-ary operations $\Delta^k : H(BR_n)^{\otimes k}$ for $3 \leq k < n$, and*
- *a non-trivial n-ary operation $\Delta^n : H(BR) \to H(BR)^{\otimes n}$.*

Predictions

Hence, for the Borromean rings we are expected to find:

- a primitive Δ^2
- a non-trivial Δ^3

These coproducts will be induced through the Transfer Algorithm.

Table of Contents

Transferring Coproducts

Goal:

$$\begin{array}{c} A_\infty\text{-coalgebra on chains} \\ (C, \partial, \Delta_2, \Delta_3, \ldots) \\ \downarrow \\ (H, 0, \Delta^2, \Delta^3, \ldots) \\ A_\infty\text{-coalgebra in homology} \end{array}$$

Transferring Coproducts

Required input:

- Coalgebra on chains $(C, \partial, \Delta_2, \Delta_3, \ldots)$ and
- a cycle-selecting map $g : H \to Z(C)$, where $Z(C)$ denotes the subspace of cycles in C.

Note: In practice we only required Δ_2 at the outset and computed the rest as needed.

How Does It Work?

Strategy: Construct a chain map from the top dimension and codim-1 cells of the $(n-1)$-dimensional multiplihedron, denoted J_n, to maps between H and $C^{\otimes n}$.

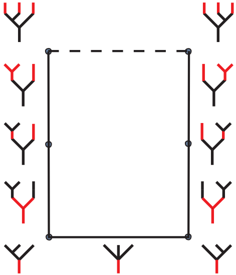
Beginning Steps

- J_n is a polytope that captures the combinatiorial structure of mapping between two A_∞-coalgebras.

Beginning Steps

- J_n is a polytope that captures the combinatiorial structure of mapping between two A_∞-coalgebras.
- Consider J_1 and J_2.

Extending to J_3



$$\mapsto$$

$$g^{\otimes 3}\left(\Delta^2 \otimes \mathbf{1}\right)\Delta^2 \qquad g^{\otimes 3}\left(\mathbf{1} \otimes \Delta^2\right)\Delta^2$$

$$\left(g^2 \otimes g\right)\Delta^2 \qquad \left(g \otimes g^2\right)\Delta^2$$

$$\left(\Delta_2 g \otimes g\right)\Delta^2 \qquad \left(g \otimes \Delta_2 g\right)\Delta^2$$

$$\left(\Delta_2 \otimes \mathbf{1}\right)g^2 \qquad \left(\mathbf{1} \otimes \Delta_2\right)g^2$$

$$\left(\Delta_2 \otimes \mathbf{1}\right)\Delta_2 g \qquad \Delta_3 g \qquad \left(\mathbf{1} \otimes \Delta_2\right)\Delta_2 g$$

Table of Contents

Linear Algebraic Methods

Good News

Linear algebra provides robust and theoretically correct methods for solving the various induction steps of the transfer algorithm.

Linear Algebraic Methods

Good News

Linear algebra provides robust and theoretically correct methods for solving the various induction steps of the transfer algorithm.

Bad News

The matrices are too large to be solved within a reasonable amount of storage space and time.

Two Problems

Problem (Preboundary)

Given a cycle $x \in C^{\otimes n}$ *of degree* k*, find a chain* $y \in C^{\otimes n}$ *of degree* $k+1$*, such that* $\partial(y) = x$.

Problem (Factorization)

Given a cycle $c \in Z(C^{\otimes n})$*, find all subcycles of* c *of the form* $Z(C)^{\otimes n}$.

Preboundary Problem: Δ_3

- First problem arose in computing Δ_3

Preboundary Problem: Δ_3

- First problem arose in computing Δ_3
- It is the preboundary of $(\Delta_2 \otimes 1 + 1 \otimes \Delta_2)\Delta_2$

Preboundary Problem: Δ_3

- First problem arose in computing Δ_3
- It is the preboundary of $(\Delta_2 \otimes 1 + 1 \otimes \Delta_2)\Delta_2$
- Brute force linear algebra approach entails 1.8 mil row $\times$ 4 mil column matrix

Preboundary Problem: Δ_3

- First problem arose in computing Δ_3
- It is the preboundary of $(\Delta_2 \otimes 1 + 1 \otimes \Delta_2)\Delta_2$
- Brute force linear algebra approach entails 1.8 mil row $\times$ 4 mil column matrix
- Instead, solved with a best-first search algorithm

Factorization Problem

- Second problem comes from deriving Δ^n

Factorization Problem

- Second problem comes from deriving Δ^n
- Transfer Algorithm specifies computing $[\phi_n]$, i.e., $H_*(\mathrm{Hom}(H, Z(C^{\otimes(n+2)})))$

Factorization Problem

- Second problem comes from deriving Δ^n
- Transfer Algorithm specifies computing $[\phi_n]$, i.e., $H_*(\mathrm{Hom}(H, Z(C^{\otimes(n+2)})))$
- However, Künneth Theorem tells us that $H_*(C^{\otimes n}) \cong H_*(C)^{\otimes n}$

Factorization Problem

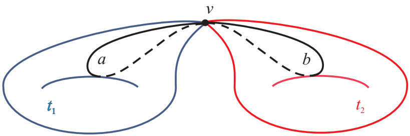
- Second problem comes from deriving Δ^n
- Transfer Algorithm specifies computing $[\phi_n]$, i.e., $H_*(\mathrm{Hom}(H, Z(C^{\otimes(n+2)})))$
- However, Künneth Theorem tells us that $H_*(C^{\otimes n}) \cong H_*(C)^{\otimes n}$
- Hence, non-boundary cycles in ϕ_n in should be of the form $Z(C)^{\otimes(n+2)}$

Factorization Problem

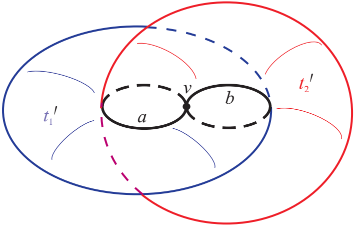
- Second problem comes from deriving Δ^n
- Transfer Algorithm specifies computing $[\phi_n]$, i.e., $H_*(\mathrm{Hom}(H, Z(C^{\otimes(n+2)})))$
- However, Künneth Theorem tells us that $H_*(C^{\otimes n}) \cong H_*(C)^{\otimes n}$
- Hence, non-boundary cycles in ϕ_n in should be of the form $Z(C)^{\otimes(n+2)}$
- Again, an algorithmic approach appears to be a feasible alternative

Table of Contents

Unlink vs. Hopf Link

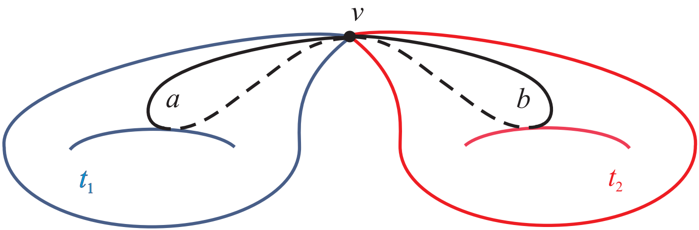


2-Component Unlink



Hopf Link

2-Component Unlink



Hopf Link

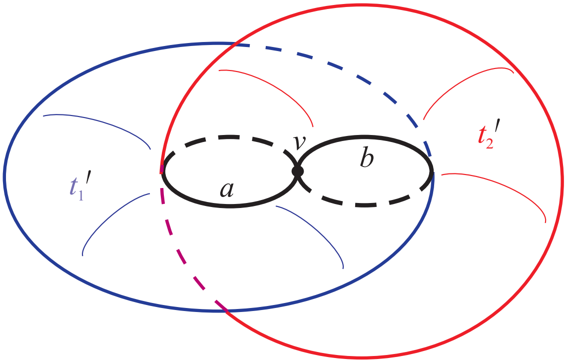


Table of Contents

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink
- a non-primitive Δ^2 for the Hopf

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink
- a non-primitive Δ^2 for the Hopf
- for the Borromean rings

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink
- a non-primitive Δ^2 for the Hopf
- for the Borromean rings
  - a primitive Δ^2

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink
- a non-primitive Δ^2 for the Hopf
- for the Borromean rings
  - a primitive Δ^2
  - a non-trivial Δ^3

Results

Transfer algorithm yields

- a primitive Δ^2 for unlink
- a non-primitive Δ^2 for the Hopf
- for the Borromean rings
    - a primitive Δ^2
    - a non-trivial Δ^3
- all of which are consistent with the conjecture!

Future Work

- ϕ_2 was non-trivial in BR_3, so either Δ^4 or g^4 is non-trivial (or both)

Future Work

- ϕ_2 was non-trivial in BR_3, so either Δ^4 or g^4 is non-trivial (or both)
- the 4-component Brunnian link is a natural next step, BUT...

Future Work

- ϕ_2 was non-trivial in BR_3, so either Δ^4 or g^4 is non-trivial (or both)
- the 4-component Brunnian link is a natural next step, BUT...
    - Δ_3 appears significantly harder to compute

Future Work

- ϕ_2 was non-trivial in BR_3, so either Δ^4 or g^4 is non-trivial (or both)
- the 4-component Brunnian link is a natural next step, BUT...
  - Δ_3 appears significantly harder to compute
  - the last steps of BR_3 were actually done by hand, and BR_4 will only be worse

Future Work

- ϕ_2 was non-trivial in BR_3, so either Δ^4 or g^4 is non-trivial (or both)
- the 4-component Brunnian link is a natural next step, BUT...
    - Δ_3 appears significantly harder to compute
    - the last steps of BR_3 were actually done by hand, and BR_4 will only be worse
- both the preboundary and factorization algorithms need improvement

Thank You

Thank You!