Detecting Linkage in an n-Component Brunnian Link

(work in progress)

Ron Umble, Barbara Nimershiem, and Merv Fansler
Millersville U and Franklin & Marshall College

Tetrahedral Geometry/Topology Seminar

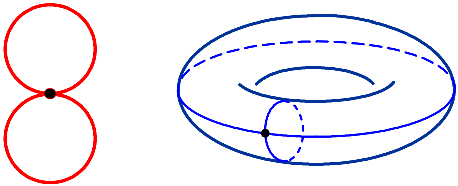
December 4, 2015

Ultimate Goal of the Project

Computationally detect the linkage in an n-component Brunnian link

Review of Cellular Complexes

Let X be a connected network, surface, or solid embedded in S^3



Cellular Decompositions

A **cellular decomposition** of X is a finite collection of

- discrete points (vertices or 0-cells)

Cellular Decompositions

A **cellular decomposition** of X is a finite collection of

- discrete points (vertices or 0-cells)
- closed intervals (edges or 1-cells)

Cellular Decompositions

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 Decompositions

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 Decompositions

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 Decompositions

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 Decompositions

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)

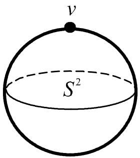
glued together so that the

- 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

Example: 2-dim'l Sphere

$S^2 = D^2/\partial D^2$ (Grandma's draw string bag)

- Vertex: $\{v\}$
- Edges: $\varnothing$
- Face: $\{S^2\}$

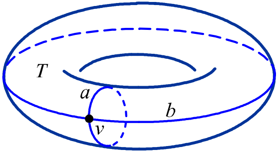


Example: Torus

$T = S^1 \times S^1$

Product cells: $\{v, a\} \times \{v, b\}$

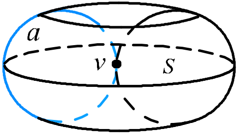
- Vertex: $\{v := v \times v\}$
- Edges: $\{a := a \times v,\ b := v \times b\}$
- Face: $\{T := a \times b\}$



Example: Pinched Sphere

$P = T/b$

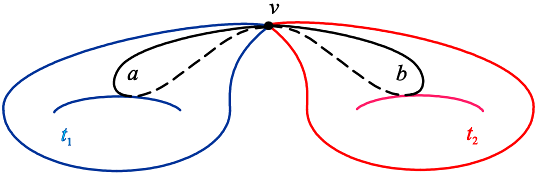
- Vertex: $\{v\}$
- Edge: $\{a\}$
- Face: $\{S\}$



Example: Link Complement of Two Unknots

Let UN be the complement of disjoint tubular neighborhoods U_i of **two unlinked unknots in** S^3

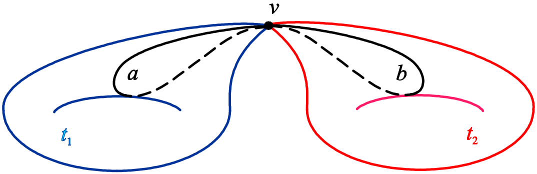
- $\partial\left(U_1 \cup U_2\right)$ is the wedge of two pinched spheres t_i sharing a single vertex v and two edges a and b



$$\partial\left(U_1 \cup U_2\right) = \partial\left(UN\right)$$

Cellular Structure of UN

- $\partial(UN)$ is wedged with the equatorial 2-sphere $s \subset S^3$
- $p =$ upper hemispherical 3-ball
- $q =$ lower hemispherical 3-ball $\setminus (U_1 \cup U_2)$
- p and q are attached along s
- $UN = p \cup q$

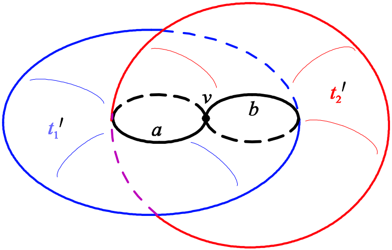


- Vertices: $\{v\}$
- Edges: $\{a, b\}$
- Faces: $\{s, t_1, t_2\}$
- Solids: $\{p, q\}$

Example: Link Complement of the Hopf Link

Let LN be the complement of disjoint tubular neighborhoods U_i of the **Hopf Link** in S^3

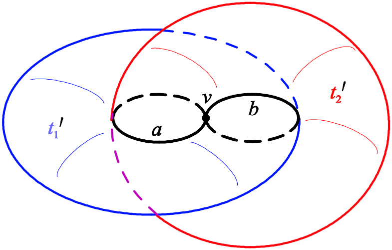
- $\partial\left(U_1 \cup U_2\right)$ is the union of two linked tori t_i' sharing a single vertex v and two edges a and b



$$\partial\left(U_1 \cup U_2\right) = \partial\left(LN\right)$$

Cellular Structure of LN

- $\partial(LN)$ is wedged with the equatorial 2-sphere $s \subset S^3$
- $p =$ upper hemispherical 3-ball
- $q' =$ lower hemispherical 3-ball $\setminus (U_1 \cup U_2)$
- p and q' are attached along s
- $LN = p \cup q'$



- Vertex: $\{v\}$
- Edges: $\{a, b\}$
- Faces: $\{s, t_1', t_2'\}$
- Solids: $\{p, q'\}$

Homeomorphisms

X and Y are **homeomorphic** if

- X can be continuously deformed into Y

Homeomorphisms

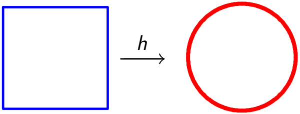
X and Y are **homeomorphic** if

- X can be continuously deformed into Y
- $\exists$ a bijective bicontinuous $h : X \to Y$, called a **homeomorphism**

Homeomorphisms

X and Y are **homeomorphic** if

- X can be continuously deformed into Y
- $\exists$ a bijective bicontinuous $h : X \to Y$, called a **homeomorphism**
- A square and a circle are homeomorphic

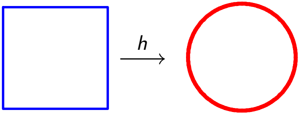


Homeomorphisms

X and Y are **homeomorphic** if

- X can be continuously deformed into Y
- $\exists$ a bijective bicontinuous $h : X \to Y$, called a **homeomorphism**

- A square and a circle are homeomorphic



- The boundaries of a doughnut and coffee cup are homeomorphic

Homeomorphisms

- UN and LN are not homeomorphic because...

Homeomorphisms

- *UN* and *LN* are not homeomorphic because...
- Shrinking the tubular neighborhood of one component to point

Homeomorphisms

- UN and LN are not homeomorphic because...
- Shrinking the tubular neighborhood of one component to point
  - shrinks $\partial(UN)$ to a pinched sphere

Homeomorphisms

- UN and LN are not homeomorphic because...
- Shrinking the tubular neighborhood of one component to point
  - shrinks $\partial\left(UN\right)$ to a pinched sphere
  - shrinks $\partial\left(LN\right)$ to a 2-sphere

Homeomorphisms

- UN and LN are not homeomorphic because...
- Shrinking the tubular neighborhood of one component to point
  - shrinks $\partial(UN)$ to a pinched sphere
  - shrinks $\partial(LN)$ to a 2-sphere
- How do we can detect this computationally?

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$
- A homeomorphism $h : X \to Y$ respects diagonals

$$\Delta_Y h = (h \times h)\,\Delta_X$$

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$
- A homeomorphism $h : X \to Y$ respects diagonals

$$\Delta_Y h = (h \times h)\,\Delta_X$$

- **Objective:** Compute the obstruction to a homeomorphism

$$h : UN \to LN$$

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$
- A homeomorphism $h : X \to Y$ respects diagonals

$$\Delta_Y h = (h \times h)\,\Delta_X$$

- **Objective:** Compute the obstruction to a homeomorphism

$$h : UN \to LN$$

- **Strategy:**

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$
- A homeomorphism $h : X \to Y$ respects diagonals

$$\Delta_Y h = (h \times h)\,\Delta_X$$

- **Objective:** Compute the obstruction to a homeomorphism

$$h : UN \to LN$$

- **Strategy:**
  - Assume there is a homeomorphism h

The Geometric Diagonal

- **Geometric diagonal** $\Delta_X : X \to X \times X$ is defined $x \mapsto (x, x)$
- A homeomorphism $h : X \to Y$ respects diagonals

$$\Delta_Y h = (h \times h)\,\Delta_X$$

- **Objective:** Compute the obstruction to a homeomorphism

$$h : UN \to LN$$

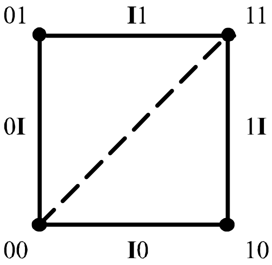
- **Strategy:**
  - Assume there is a homeomorphism h
  - Show that h fails to respect diagonals

The Geometric Diagonal

- **Problem:** $\operatorname{Im}\Delta_X$ is typically *not* a subcomplex of $X \times X$

The Geometric Diagonal

- **Problem:** $\mathrm{Im}\,\Delta_X$ is typically *not* a subcomplex of $X \times X$
- **Example:** $\mathrm{Im}\,\Delta_{\mathtt{I}}$ is not a subcomplex of $\mathtt{I} \times \mathtt{I}$:



Diagonal Approximations

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

Diagonal Approximations

- A map $\Delta : X \rightarrow X \times X$ is a **diagonal approximation** if
  - Δ is homotopic to Δ_X

Diagonal Approximations

- A map $\Delta : X \to X \times X$ is a **diagonal approximation** if
  - Δ is homotopic to Δ_X
  - $\Delta\left(e^n\right)$ is a subcomplex of $e^n \times e^n$ for every n-cell $e^n \subseteq X$

Diagonal Approximations

- A map $\Delta : X \to X \times X$ is a **diagonal approximation** if
  - Δ is homotopic to Δ_X
  - $\Delta(e^n)$ is a subcomplex of $e^n \times e^n$ for every n-cell $e^n \subseteq X$
  - Geometric boundary $\partial : X \to X$ is a coderivation of Δ

$$\Delta\partial = (\partial \times \mathrm{Id} + \mathrm{Id} \times \partial)\,\Delta$$

Diagonal Approximations

- A map $\Delta : X \to X \times X$ is a **diagonal approximation** if
  - Δ is homotopic to Δ_X
  - $\Delta(e^n)$ is a subcomplex of $e^n \times e^n$ for every n-cell $e^n \subseteq X$
  - Geometric boundary $\partial : X \to X$ is a coderivation of Δ

$$\Delta\partial = (\partial \times \mathrm{Id} + \mathrm{Id} \times \partial)\,\Delta$$

- **Cellular Approximation Theorem**

There is a diagonal approximation $\Delta : X \to X \times X$

Diagonal Approximations

Preserve

- Cellular structure: $\Delta\left(e^n\right) \subseteq e^n \times e^n$

Diagonal Approximations

Preserve

- Cellular structure: $\Delta\left(e^n\right) \subseteq e^n \times e^n$
- Dimension: $\dim \Delta\left(e^n\right) = \dim e^n$

Diagonal Approximations

Preserve

- Cellular structure: $\Delta\left(e^n\right) \subseteq e^n \times e^n$
- Dimension: $\dim \Delta\left(e^n\right) = \dim e^n$
- Cartesian products: $\Delta\left(X \times Y\right) = \Delta\left(X\right) \times \Delta\left(Y\right)$

Diagonal Approximations

Preserve

- Cellular structure: $\Delta\left(e^n\right) \subseteq e^n \times e^n$
- Dimension: $\dim \Delta\left(e^n\right) = \dim e^n$
- Cartesian products: $\Delta\left(X \times Y\right) = \Delta\left(X\right) \times \Delta\left(Y\right)$
- Wedge products: $\Delta\left(X \vee Y\right) = \Delta\left(X\right) \vee \Delta\left(Y\right)$

Dan Kravatz's Diagonal Approximation on a Polygon

- Given n-gon G, arbitrarily choose vertices v and v' (possibly equal)

Dan Kravatz's Diagonal Approximation on a Polygon

- Given n-gon G, arbitrarily choose vertices v and v' (possibly equal)
- Edges $\{e_1, \ldots, e_k\}$ and $\{e_{k+1}, \ldots, e_n\}$ form paths from v to v' (one path $\{e_1, \ldots, e_n\}$ if $v = v'$)

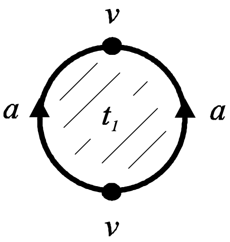
Dan Kravatz's Diagonal Approximation on a Polygon

- Given n-gon G, arbitrarily choose vertices v and v' (possibly equal)
- Edges $\{e_1, \ldots, e_k\}$ and $\{e_{k+1}, \ldots, e_n\}$ form paths from v to v' (one path $\{e_1, \ldots, e_n\}$ if $v = v'$)
- **Theorem (Kravatz 2008):** *There is a diagonal approximation*

$$\Delta G = v \times G + G \times v' + \sum_{i=2}^{k} (e_1 + \cdots + e_{i-1}) \times e_i + \sum_{j=k+2}^{n} (e_{k+1} + \cdots + e_{j-1}) \times e_j$$

Example

Think of the **pinched sphere** $t_1 \subset \partial\left(UN\right)$ as a 2-gon with vertices identified first, then edges identified

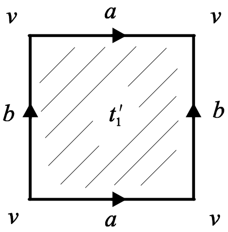


$$\Delta t_1 = v \times t_1 + t_1 \times v$$

- Δ descends to quotients when edge-paths are consistent with identifications

Example

Think of the **torus** $t_1' \subset \partial(LN)$ as a square with horizontal edges a identified and vertical edges b identified



$$\Delta t_1' = v \times t_1' + t_1' \times v + a \times b + b \times a$$

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$
  - Elements are formal sums called **cellular chains of** X

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$
  - Elements are formal sums called **cellular chains of** X
- **Examples:**

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$
  - Elements are formal sums called **cellular chains of X**
- **Examples:**
  - $C(UN)$ has basis $\{v, a, b, s, t_1, t_2, p, q\}$

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$
  - Elements are formal sums called **cellular chains of** X
- **Examples:**
  - $C(UN)$ has basis $\{v, a, b, s, t_1, t_2, p, q\}$
  - $C(LN)$ has basis $\{v, a, b, s, t_1', t_2', p, q'\}$

Cellular Chains

- $C(X)$ denotes the $\mathbb{Z}_2$-vector space with basis $\{\text{cells of } X\}$
  - Elements are formal sums called **cellular chains of** X
- **Examples:**
  - $C(UN)$ has basis $\{v, a, b, s, t_1, t_2, p, q\}$
  - $C(LN)$ has basis $\{v, a, b, s, t_1', t_2', p, q'\}$
  - Note that $C(UN) \approx C(LN)$

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$

The Boundary Operator

- **Geometric boundary** of an *n*-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial\left(\partial e^n\right) = \partial S^{n-1} = \varnothing$

The Boundary Operator

- **Geometric boundary** of an *n*-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial \left(\partial e^n \right) = \partial S^{n-1} = \varnothing$
- **Boundary operator** $\partial : C\left(X\right) \to C\left(X\right)$

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial \left(\partial e^n \right) = \partial S^{n-1} = \varnothing$
- **Boundary operator** $\partial : C(X) \to C(X)$
  - Induced by the geometric boundary

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial\left(\partial e^n\right) = \partial S^{n-1} = \varnothing$
- **Boundary operator** $\partial : C\left(X\right) \rightarrow C\left(X\right)$
  - Induced by the geometric boundary
  - Zero on vertices

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial\left(\partial e^n\right) = \partial S^{n-1} = \varnothing$
- **Boundary operator** $\partial : C\left(X\right) \to C\left(X\right)$
  - Induced by the geometric boundary
  - Zero on vertices
  - Linear on chains

The Boundary Operator

- **Geometric boundary** of an n-cell e^n is S^{n-1}
  - $\partial \mathbf{v} = \varnothing$; $\partial \mathbf{e} = S^0$; $\partial \mathbf{f} = S^1$; $\partial \mathbf{s} = S^2$
  - $\partial \left(\partial e^n \right) = \partial S^{n-1} = \varnothing$
- **Boundary operator** $\partial : C\left(X\right) \rightarrow C\left(X\right)$
  - Induced by the geometric boundary
  - Zero on vertices
  - Linear on chains
  - A derivation of the Cartesian product

$$\partial \left(a \times b \right) = \partial a \times b + a \times \partial b$$

Examples

- $\partial : C\left(UN\right) \to C\left(UN\right)$ is defined

$$\begin{aligned}
&\partial v = \partial a = \partial b = \partial s = \partial t_1 = \partial t_2 = 0\\
&\partial p = s\\
&\partial q = s + t_1 + t_2
\end{aligned}$$

Examples

- $\partial : C(UN) \to C(UN)$ is defined

$$\begin{aligned} \partial v &= \partial a = \partial b = \partial s = \partial t_1 = \partial t_2 = 0 \\ \partial p &= s \\ \partial q &= s + t_1 + t_2 \end{aligned}$$

- $\partial : C(LN) \to C(LN)$ is defined

$$\begin{aligned} \partial v &= \partial a = \partial b = \partial s = \partial t_1' = \partial t_2' = 0 \\ \partial p &= s \\ \partial q' &= s + t_1' + t_2' \end{aligned}$$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\mathrm{Im}\,\partial \subseteq \ker \partial$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\operatorname{Im} \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \operatorname{Im} \partial$ is the **cellular homology of** X

Cellular Homology

- $\partial \circ \partial = 0$ implies $\operatorname{Im} \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \operatorname{Im} \partial$ is the **cellular homology of X**
- Elements of $H(X)$ are cosets $[c] := c + \operatorname{Im} \partial$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\mathrm{Im}\, \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \mathrm{Im}\, \partial$ is the **cellular homology of** X
- Elements of $H(X)$ are cosets $[c] := c + \mathrm{Im}\, \partial$
- **Examples**

Cellular Homology

- $\partial \circ \partial = 0$ implies $\operatorname{Im} \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \operatorname{Im} \partial$ is the **cellular homology of** X
- Elements of $H(X)$ are cosets $[c] := c + \operatorname{Im} \partial$
- **Examples**
  - $H(UN) = \{[v], [a], [b], [t_1] = [t_2]\}$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\operatorname{Im} \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \operatorname{Im} \partial$ is the **cellular homology of** X
- Elements of $H(X)$ are cosets $[c] := c + \operatorname{Im} \partial$
- **Examples**
  - $H(UN) = \{[v], [a], [b], [t_1] = [t_2]\}$
  - $H(LN) = \{[v], [a], [b], [t'_1] = [t'_2]\}$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\mathrm{Im}\,\partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \mathrm{Im}\,\partial$ is the **cellular homology of** X
- Elements of $H(X)$ are cosets $[c] := c + \mathrm{Im}\,\partial$
- **Examples**
  - $H(UN) = \{[v], [a], [b], [t_1] = [t_2]\}$
  - $H(LN) = \{[v], [a], [b], [t_1'] = [t_2']\}$
  - Note that $H(UN) \approx H(LN)$

Cellular Homology

- $\partial \circ \partial = 0$ implies $\operatorname{Im} \partial \subseteq \ker \partial$
- $H(X) := \ker \partial / \operatorname{Im} \partial$ is the **cellular homology of** X
- Elements of $H(X)$ are cosets $[c] := c + \operatorname{Im} \partial$
- **Examples**
  - $H(UN) = \{[v], [a], [b], [t_1] = [t_2]\}$
  - $H(LN) = \{[v], [a], [b], [t_1'] = [t_2']\}$
  - Note that $H(UN) \approx H(LN)$
- How do diagonal approximations on UN and LN lift to homology?

Key Facts

- Homotopic maps of spaces induce *the same map* on their homologies

Key Facts

- Homotopic maps of spaces induce *the same map* on their homologies
- Every diagonal approximation $\Delta : X \to X \times X$ induces the same map

$$\Delta_2 : H(X) \to H(X \times X)$$

Key Facts

- Homotopic maps of spaces induce *the same map* on their homologies
- Every diagonal approximation $\Delta : X \to X \times X$ induces the same map

$$\Delta_2 : H(X) \to H(X \times X)$$

- A homeomorphism $h : X \to Y$ induces maps

$$h_* : H(X) \to H(Y) \text{ and } (h \times h)_* : H(X \times X) \to H(Y \times Y)$$

such that $\Delta_2 h_* = (h \times h)_* \Delta_2$

Key Facts

- Homotopic maps of spaces induce *the same map* on their homologies
- Every diagonal approximation $\Delta : X \to X \times X$ induces the same map

$$\Delta_2 : H(X) \to H(X \times X)$$

- A homeomorphism $h : X \to Y$ induces maps

$$h_* : H(X) \to H(Y) \text{ and } (h \times h)_* : H(X \times X) \to H(Y \times Y)$$

  such that $\Delta_2 h_* = (h \times h)_* \Delta_2$

- If $h : UN \to LN$ is a homeomorphism, inequality is a contradiction

Homology of Cartesian Products

- If vector space A has basis $\{a_1, \ldots, a_k\}$, the **tensor product** vector space $A \otimes A$ has basis $\{a_i \otimes a_j\}_{1 \leq i,j \leq k}$

Homology of Cartesian Products

- If vector space A has basis $\{a_1, \ldots, a_k\}$, the **tensor product** vector space $A \otimes A$ has basis $\{a_i \otimes a_j\}_{1 \leq i,j \leq k}$
- $C(X \times X) \approx C(X) \otimes C(X)$ via $e \times e' \mapsto e \otimes e'$

Homology of Cartesian Products

- If vector space A has basis $\{a_1, \ldots, a_k\}$, the **tensor product** vector space $A \otimes A$ has basis $\{a_i \otimes a_j\}_{1 \leq i,j \leq k}$
- $C(X \times X) \approx C(X) \otimes C(X)$ via $e \times e' \mapsto e \otimes e'$
- The boundary map

$$\partial \times \mathrm{Id} + \mathrm{Id} \times \partial : X \times X \to X \times X$$

induces the boundary operator

$$\partial \otimes \mathrm{Id} + \mathrm{Id} \otimes \partial : C(X) \otimes C(X) \to C(X) \otimes C(X)$$

Homology of Cartesian Products

- If vector space A has basis $\{a_1, \ldots, a_k\}$, the **tensor product** vector space $A \otimes A$ has basis $\{a_i \otimes a_j\}_{1 \leq i,j \leq k}$
- $C(X \times X) \approx C(X) \otimes C(X)$ via $e \times e' \mapsto e \otimes e'$
- The boundary map

$$\partial \times \mathrm{Id} + \mathrm{Id} \times \partial : X \times X \to X \times X$$

induces the boundary operator

$$\partial \otimes \mathrm{Id} + \mathrm{Id} \otimes \partial : C(X) \otimes C(X) \to C(X) \otimes C(X)$$

- Since $\mathbb{Z}_2$ is a field, torsion vanishes and

$$H(X \times X) \approx H(X) \otimes H(X)$$

Induced Diagonal on H(X)

- A diagonal approximation $\Delta : X \to X \times X$ induces a **coproduct**

$$\Delta_2 : H(X) \to H(X) \otimes H(X)$$

defined by

$$\Delta_2 [c] := [\Delta c]$$

Induced Diagonal on H(X)

- A diagonal approximation $\Delta : X \to X \times X$ induces a **coproduct**

$$\Delta_2 : H(X) \to H(X) \otimes H(X)$$

  defined by

$$\Delta_2 [c] := [\Delta c]$$

- A class $[c]$ of positive dimension is **primitive** if

$$\Delta_2 [c] = [v] \otimes [c] + [c] \otimes [v]$$

Induced Diagonal on H(X)

- A diagonal approximation $\Delta : X \to X \times X$ induces a **coproduct**

$$\Delta_2 : H(X) \to H(X) \otimes H(X)$$

defined by

$$\Delta_2 [c] := [\Delta c]$$

- A class $[c]$ of positive dimension is **primitive** if

$$\Delta_2 [c] = [v] \otimes [c] + [c] \otimes [v]$$

- **Examples**

$$\Delta_2 [t_1] = [\Delta t_1] = [v] \otimes [t_1] + [t_1] \otimes [v]$$

$$\Delta_2 [t_1'] = [\Delta t_1'] = [v] \otimes [t_1'] + [t_1'] \otimes [v] + [a] \otimes [b] + [b] \otimes [a]$$

Non-Primitivity Detects Linkage

- If $h : UN \to LN$ is a homeomorphism, $(h_* \otimes h_*)\Delta_2 = \Delta_2 h_*$

Non-Primitivity Detects Linkage

- If $h : UN \longrightarrow LN$ is a homeomorphism, $(h_* \otimes h_*) \Delta_2 = \Delta_2 h_*$
- But $h_* [t_1] = [t_1']$ implies

$$
\begin{aligned}
(h_* \otimes h_*) \Delta_2 [t_1] &= (h_* \otimes h_*) ([v] \otimes [t_1] + [t_1] \otimes [v]) \\
&= [v] \otimes [t_1'] + [t_1'] \otimes [v] \\
&\neq [v] \otimes [t_1'] + [t_1'] \otimes [v] + [a] \otimes [b] + [b] \otimes [a] \\
&= \Delta_2 [t_1'] = \Delta_2 h_* [t_1],
\end{aligned}
$$

which is a contradiction

Non-Primitivity Detects Linkage

- If $h : UN \longrightarrow LN$ is a homeomorphism, $(h_* \otimes h_*)\,\Delta_2 = \Delta_2 h_*$
- But $h_*\,[t_1] = [t_1']$ implies

$$
\begin{aligned}
(h_* \otimes h_*)\,\Delta_2\,[t_1] &= (h_* \otimes h_*)\,([v] \otimes [t_1] + [t_1] \otimes [v]) \\
&= [v] \otimes [t_1'] + [t_1'] \otimes [v] \\
&\neq [v] \otimes [t_1'] + [t_1'] \otimes [v] + [a] \otimes [b] + [b] \otimes [a] \\
&= \Delta_2\,[t_1'] = \Delta_2 h_*\,[t_1]\,,
\end{aligned}
$$

which is a contradiction

- **The non-primitive coproduct has detected the Hopf Link!**

Recap

- Homology alone cannot distinguish links from unlinks.

Recap

- Homology alone cannot distinguish links from unlinks.
- For example,

$$\begin{aligned} H(UN) &= \{[v], [a], [b], [t_1] = [t_2]\} \\ \text{and } H(LN) &= \{[v], [a], [b], [t_1'] = [t_2']\}, \end{aligned}$$

so $H(UN) \approx H(LN)$.

Recap

- Homology alone cannot distinguish links from unlinks.
- For example,

$$\begin{aligned} H(UN) &= \{[v],[a],[b],[t_1]=[t_2]\} \\ \text{and } H(LN) &= \{[v],[a],[b],[t_1']=[t_2']\}, \end{aligned}$$

  so $H(UN) \approx H(LN)$.
- But perhaps homology with additional structure derived from diagonal approximations can distinguish between links.

Recap

- Homology alone cannot distinguish links from unlinks.
- For example,

$$\begin{aligned} H(UN) &= \{[v], [a], [b], [t_1] = [t_2]\} \\ \text{and } H(LN) &= \{[v], [a], [b], [t_1'] = [t_2']\}, \end{aligned}$$

  so $H(UN) \approx H(LN)$.
- But perhaps homology with additional structure derived from diagonal approximations can distinguish between links.
- For example, the coproducts, Δ_2, induced by diagonal approximations are different for UN and LN.

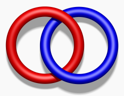
$$\begin{aligned} \Delta_2 [t_1] &= [v] \otimes [t_1] + [t_1] \otimes [v] \\ &\quad \text{is primitive.} \\ \Delta_2 [t_1'] &= [v] \otimes [t_1'] + [t_1'] \otimes [v] + [a] \otimes [b] + [b] \otimes [a] \\ &\quad \text{is not primitive.} \end{aligned}$$

Definition

- **Definition:** *A nontrivial link is called Brunnian if it has the following property: Removing any component results in an unlink.*

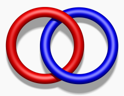
Definition

- **Definition:** *A nontrivial link is called Brunnian if it has the following property: Removing any component results in an unlink.*
- **(a non-standard) Example:** The Hopf link.

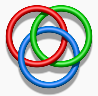


Definition

- **Definition:** *A nontrivial link is called Brunnian if it has the following property: Removing any component results in an unlink.*
- **(a non-standard) Example:** The Hopf link.

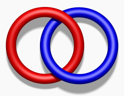


- **(the most standard) Example:** The Borromean rings.

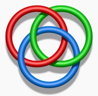


Definition

- **Definition:** *A nontrivial link is called Brunnian if it has the following property: Removing any component results in an unlink.*
- **(a non-standard) Example:** The Hopf link.



- **(the most standard) Example:** The Borromean rings.



- **Convention:** Let BR_n denote the complement in S^3 of a Brunnian link with n components where $n > 3$.

Conjecture

Conjecture: *A diagonal approximation* Δ *on* $C(BR_n)$ *induces*

- *a primitive diagonal* $\Delta_2 : H(BR_n) \rightarrow H(BR_n) \otimes H(BR_n)$,

Conjecture

Conjecture: *A diagonal approximation* Δ *on* $C(BR_n)$ *induces*

- *a primitive diagonal* $\Delta_2 : H(BR_n) \to H(BR_n) \otimes H(BR_n)$,
- *trivial k-ary operations* $\Delta_k : H(BR_n) \to H(BR_n)^{\otimes k}$ *for* $3 \leq k < n$, *and*

Conjecture

Conjecture: *A diagonal approximation* Δ *on* $C(BR_n)$ *induces*

- *a primitive diagonal* $\Delta_2 : H(BR_n) \to H(BR_n) \otimes H(BR_n)$,
- *trivial k-ary operations* $\Delta_k : H(BR_n) \to H(BR_n)^{\otimes k}$ *for* $3 \leq k < n$, *and*
- *a non-trivial n-ary operation* $\Delta_n : H(BR) \to H(BR)^{\otimes n}$.

Work to do

- Describe an infinite family of Brunnian links, including cell decompositions and diagonal approximations.

Work to do

- Describe an infinite family of Brunnian links, including cell decompositions and diagonal approximations.
- (computationally) Transfer the differential graded coalgebra structure on the chains to homology.

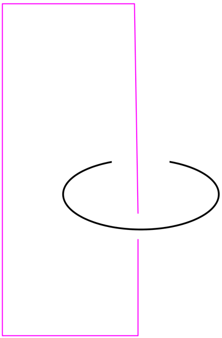
Work to do

- Describe an infinite family of Brunnian links, including cell decompositions and diagonal approximations.
- (computationally) Transfer the differential graded coalgebra structure on the chains to homology.
- (hopefully) Observe the conjectured results.

Work to do

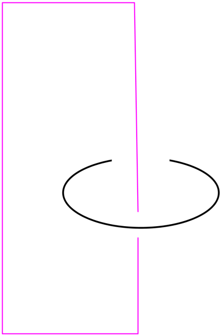
- Describe an infinite family of Brunnian links, including cell decompositions and diagonal approximations.
- (computationally) Transfer the differential graded coalgebra structure on the chains to homology.
- (hopefully) Observe the conjectured results.
- Prove them in general.

The Hopf link: A Brunnian link with two components

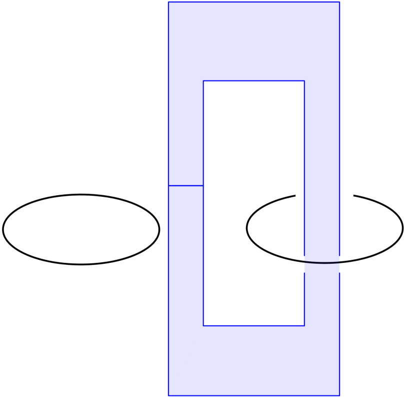


Constructing a Brunnian link with 3 components

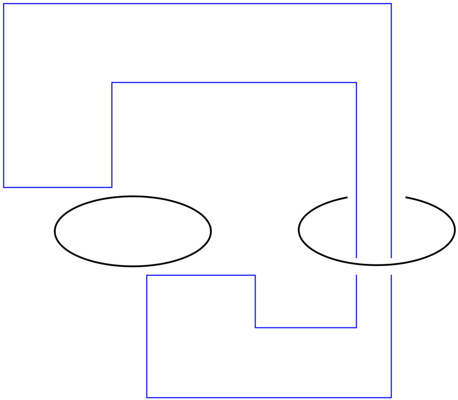
Constructing a Brunnian link with 3 components



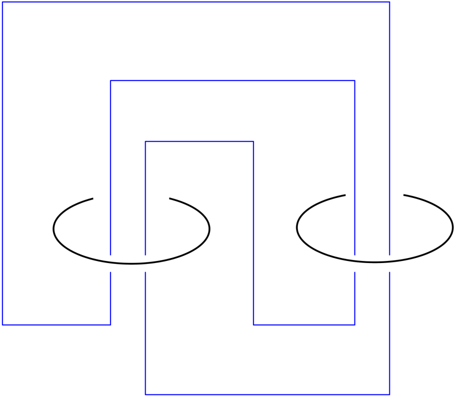
Constructing a Brunnian link with 3 components



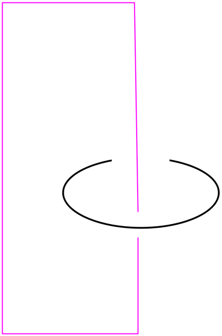
Constructing a Brunnian link with 3 components



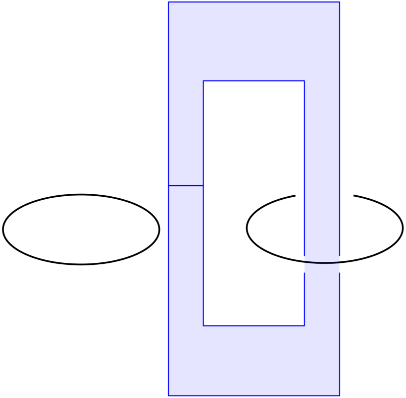
Constructing a Brunnian link with 3 components



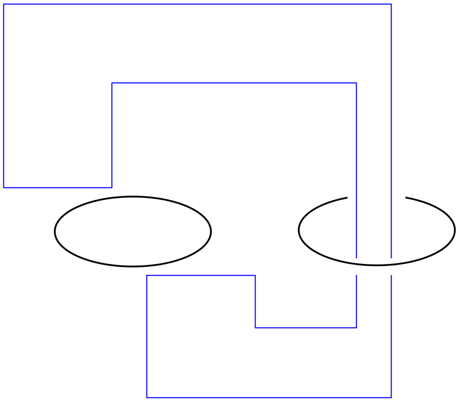
Constructing a Brunnian link with 3 components



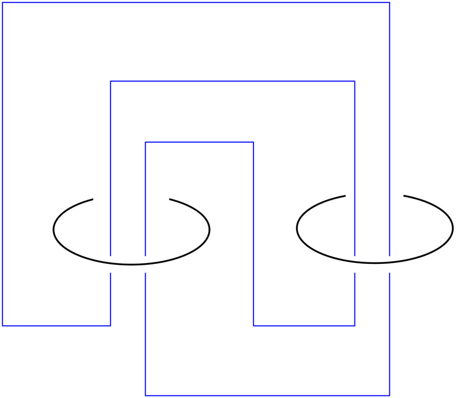
Constructing a Brunnian link with 3 components



Constructing a Brunnian link with 3 components

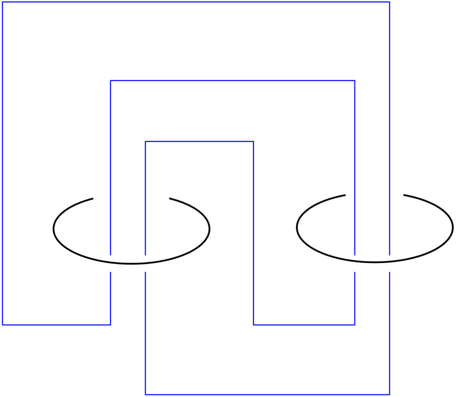


Constructing a Brunnian link with 3 components

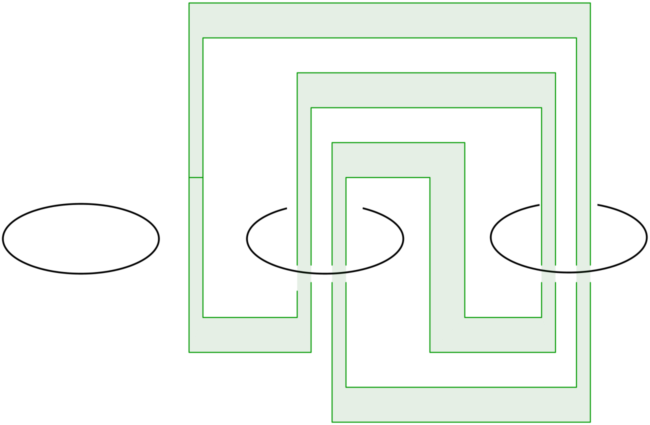


Constructing a Brunnian link with 4 components

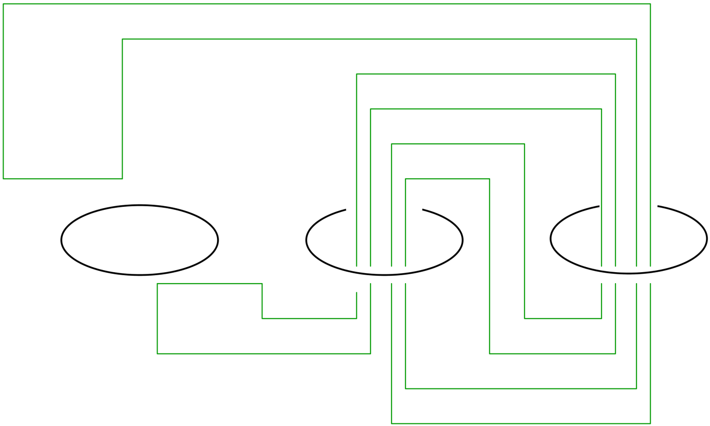
Constructing a Brunnian link with 4 components



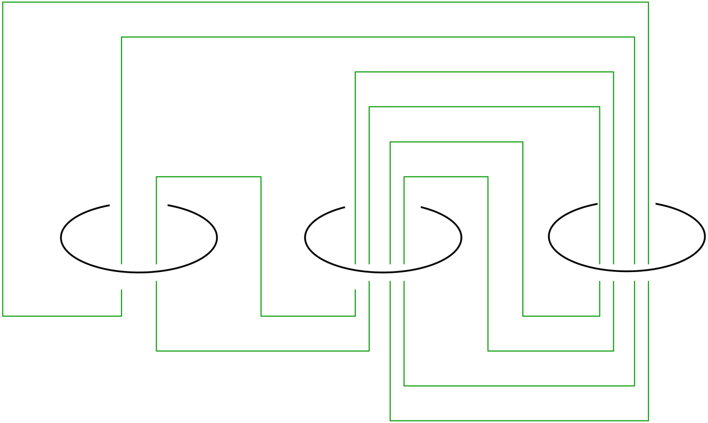
Constructing a Brunnian link with 4 components



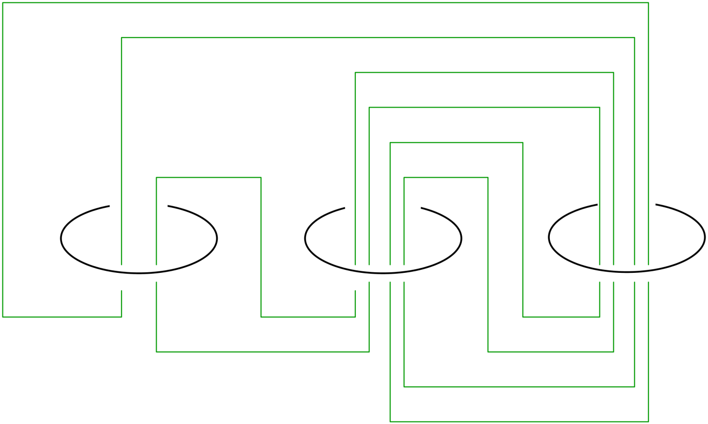
Constructing a Brunnian link with 4 components



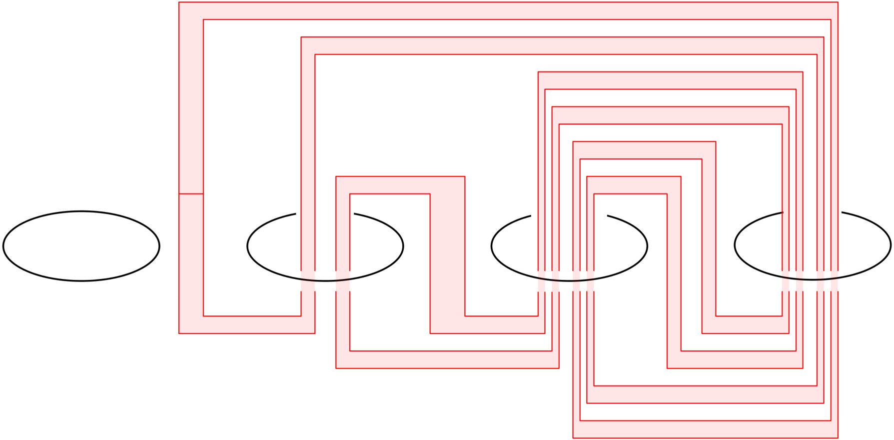
Constructing a Brunnian link with 4 components



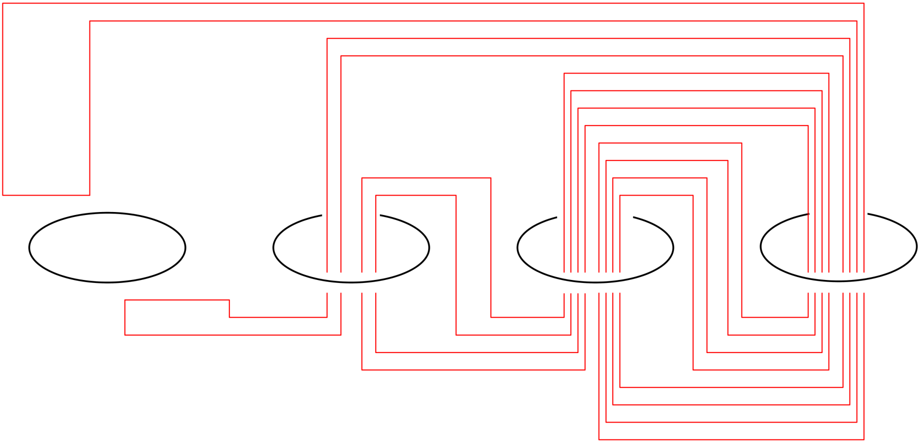
Constructing a Brunnian link with 5 components



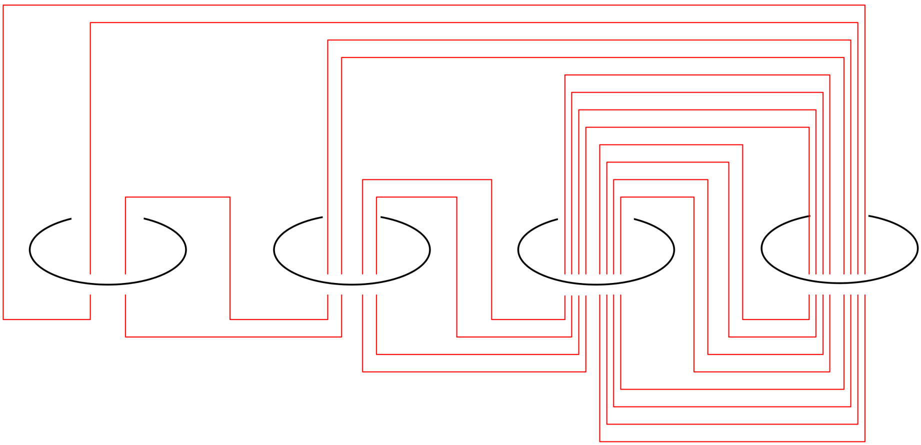
Constructing a Brunnian link with 5 components



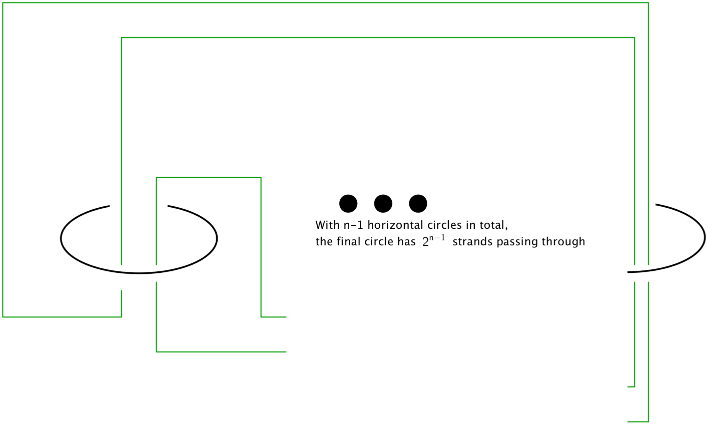
Constructing a Brunnian link with 5 components



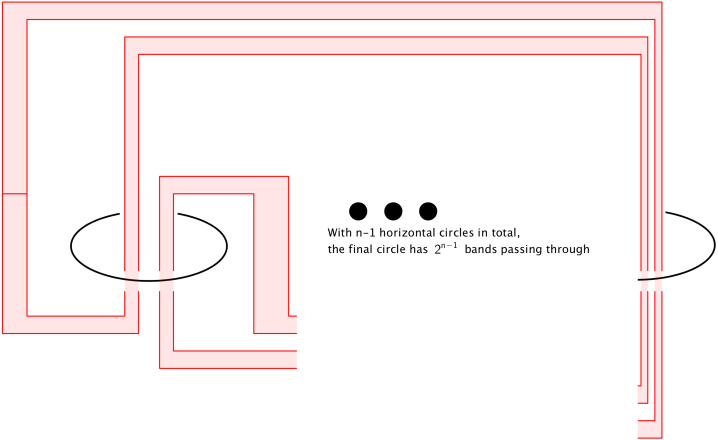
Constructing a Brunnian link with 5 components



And so on . . .

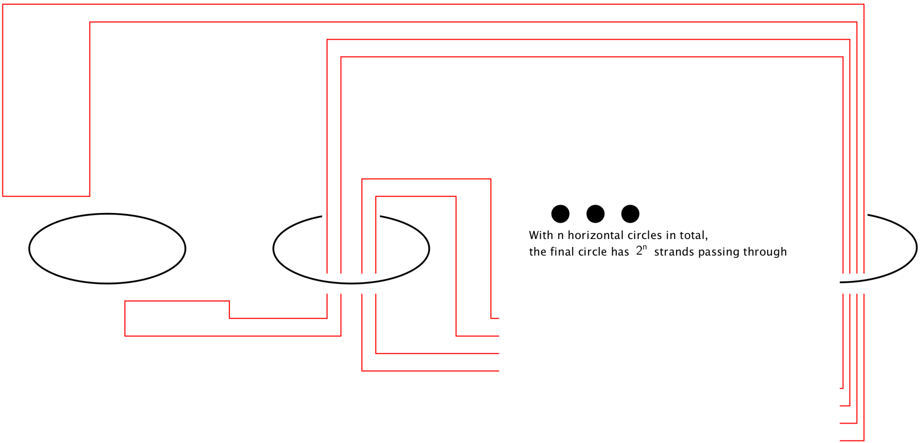

With n−1 horizontal circles in total,
the final circle has 2^{n-1} strands passing through

And so on . . .

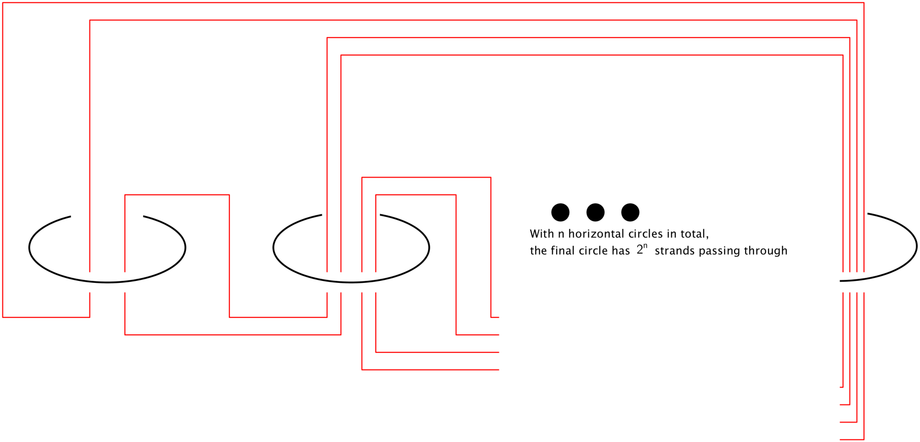


With n−1 horizontal circles in total,
the final circle has 2^{n-1} bands passing through

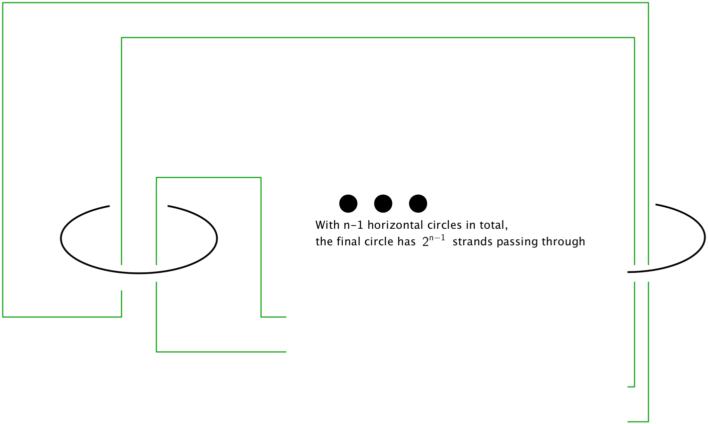
And so on . . .


With n horizontal circles in total,
the final circle has 2^n strands passing through

And so on . . .

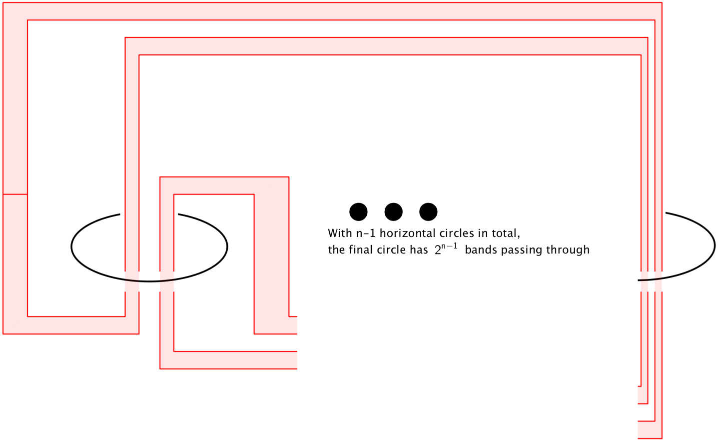

With n horizontal circles in total,
the final circle has 2^n strands passing through

Assume the constructed links are Brunnian for $k = n$

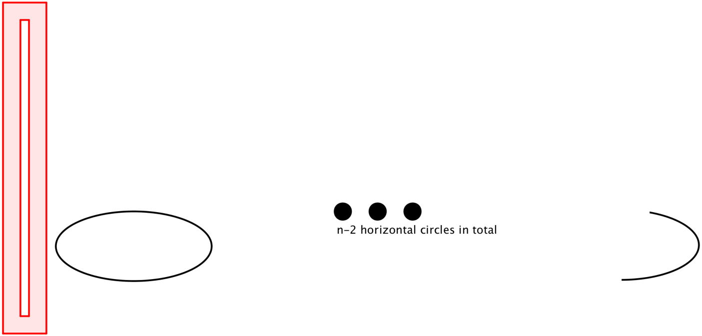


Assume the constructed links are Brunnian for $k = n$

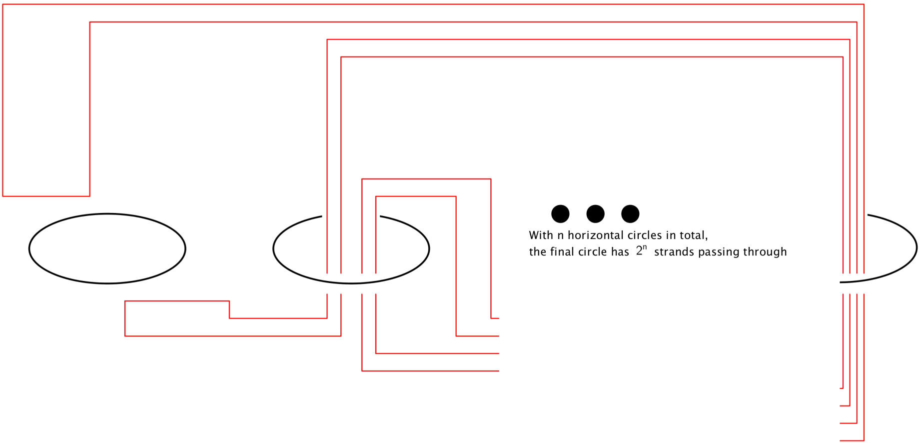
Assume the constructed links are Brunnian for $k = n$



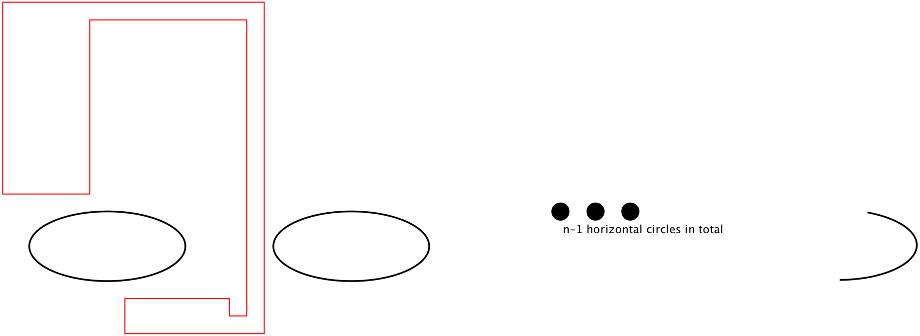
Assume the constructed links are Brunnian for $k = n$



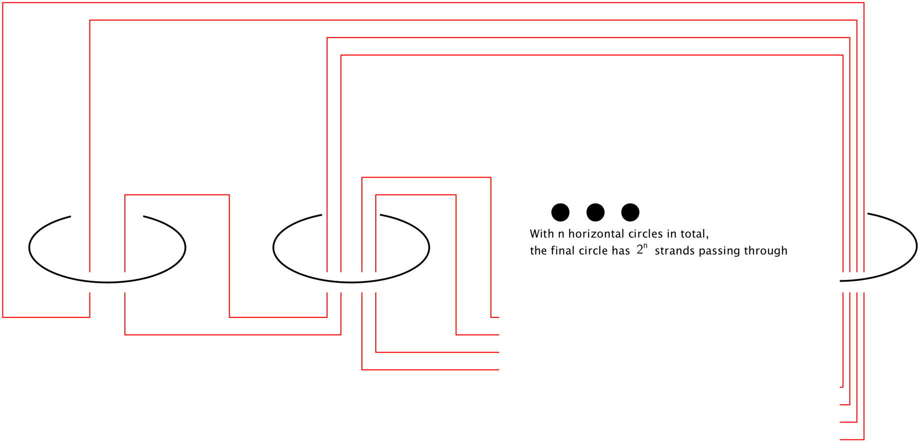
Assume the constructed links are Brunnian for $k = n$



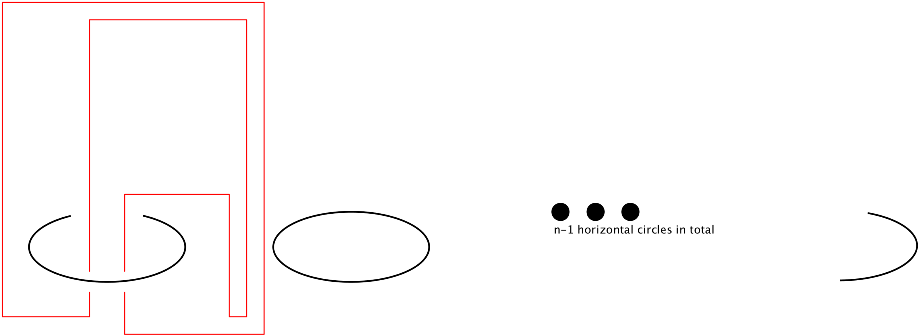
Assume the constructed links are Brunnian for $k = n$



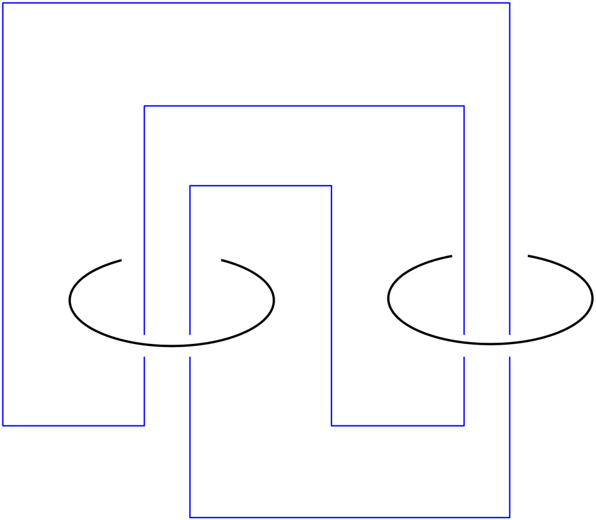
Then the constructed links are Brunnian for $k = n + 1$



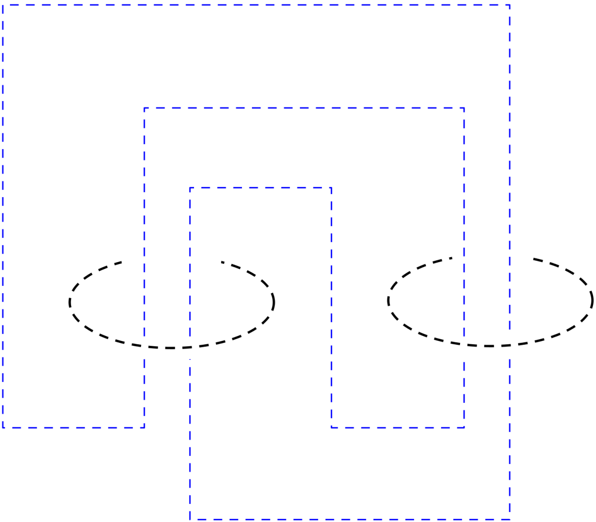
Then the constructed links are Brunnian for $k = n + 1$



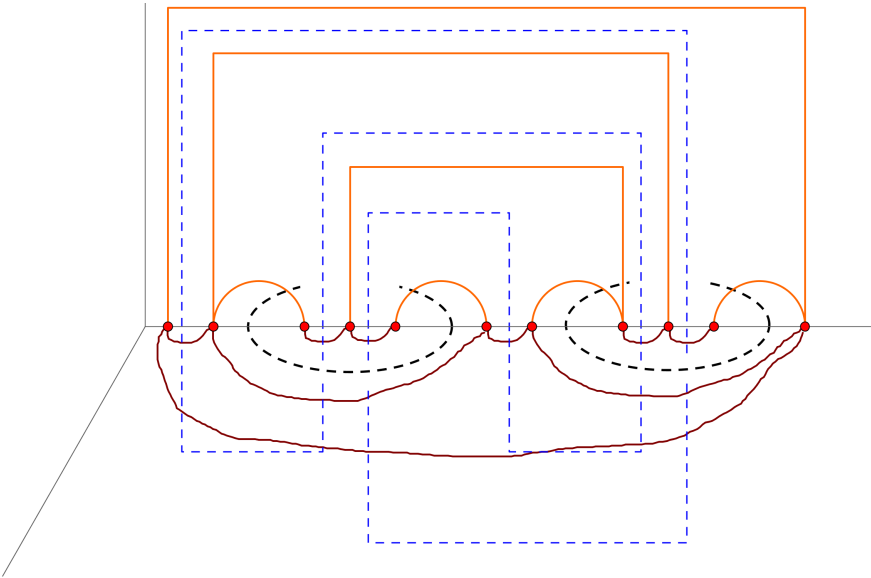
An example of a cell decomposition



An example of a cell decomposition



An example of a cell decomposition



Software Components

We require the following software components:

- parser for reading chain complex C and coproduct definition Δ_C

Software Components

We require the following software components:

- parser for reading chain complex C and coproduct definition Δ_C
- utility to validate Δ_C

Software Components

We require the following software components:

- parser for reading chain complex C and coproduct definition Δ_C
- utility to validate Δ_C
- program to compute homology (H) and a cycle-selecting map $g : H \to C$

Software Components

We require the following software components:

- parser for reading chain complex C and coproduct definition Δ_C
- utility to validate Δ_C
- program to compute homology (H) and a cycle-selecting map $g : H \to C$
- program to initialize transfer of coproduct $\Delta_C : C \to C \otimes C$ to homology $\Delta_2 : H \to H \otimes H$

Software Components

We require the following software components:

- parser for reading chain complex C and coproduct definition Δ_C
- utility to validate Δ_C
- program to compute homology (H) and a cycle-selecting map $g : H \to C$
- program to initialize transfer of coproduct $\Delta_C : C \to C \otimes C$ to homology $\Delta_2 : H \to H \otimes H$
- program to inductively compute higher coalgebraic structures in homology (Δ_n)

Complex Specification (UN)

Used a subset of LaTeX to specify chain complexes

Cells

```
% == CELLS ================================

C_{0}(UN) = \{ v \}
C_{1}(UN) = \{ a, b \}
C_{2}(UN) = \{ s, t_{1}, t_{2} \}
C_{3}(UN) = \{ p, q \}
```

Boundary

```
% == BOUNDARY ==========================

\partial p = s
\partial q = s + t_{1} + t_{2}
```

Complex Specification (UN)

Diagonals

```
% == COPRODUCT ====================================

\Delta v = v \otimes v

\Delta a = v \otimes a + a \otimes v
\Delta b = v \otimes b + b \otimes v

\Delta s = v \otimes s + s \otimes v
\Delta t_{1} = v \otimes t_{1} + t_{1} \otimes v
\Delta t_{2} = v \otimes t_{2} + t_{2} \otimes v

\Delta p = v \otimes p + p \otimes v
\Delta q = v \otimes q + q \otimes v
```

Complex Specification (LN)

For Hopf link, the cells and boundary are identical,

Cells

```
% == CELLS ============================

C_{0}(LN) = \{ v \}
C_{1}(LN) = \{ a, b \}
C_{2}(LN) = \{ s, t_{1}, t_{2} \}
C_{3}(LN) = \{ p, q \}
```

Boundary

```
% == BOUNDARY ========================

\partial p = s
\partial q = s + t_{1} + t_{2}
```

Complex Specification (LN)

But the diagonals are different,

Diagonals

```
% == COPRODUCT ================================================================

\Delta v = v \otimes v

\Delta a = v \otimes a + a \otimes v
\Delta b = v \otimes b + b \otimes v

\Delta s = v \otimes s + s \otimes v
\Delta t_{1} = v \otimes t_{1} + a \otimes b + b \otimes a + t_{1} \otimes v
\Delta t_{2} = v \otimes t_{2} + a \otimes b + b \otimes a + t_{2} \otimes v

\Delta p = v \otimes p + p \otimes v
\Delta q = v \otimes q + q \otimes v
```

Parser

- written in Python
- uses open-source library PLY (Python Lex-Yacc)
  - specify tokens
  - specify formal grammar using tokens
  - it generates LR parser
- additionally wrote utility to export *ChainComplex* objects for **SageMath**

Validating Coproduct Definition

Need to verify that $\Delta\partial = (1 \otimes \partial + \partial \otimes 1)\Delta$. We have a script that checks this for all $c \in C$.

```
Δ∂p = Δ (s)
    = s ⊗ v + v ⊗ s

(1⊗∂ + ∂⊗1)Δp = (1⊗∂ + ∂⊗1) (p ⊗ v + v ⊗ p)
              = s ⊗ v + v ⊗ s

Diagonal Valid!: Δ∂p == (1⊗∂ + ∂⊗1)Δp
Δ∂q = Δ (s + t_{2} + t_{1})
    = s ⊗ v + t_{2} ⊗ v + t_{1} ⊗ v + v ⊗ t_{2} + v ⊗ t_{1} + v ⊗ s

(1⊗∂ + ∂⊗1)Δq = (1⊗∂ + ∂⊗1) (v ⊗ q + q ⊗ v)
              = s ⊗ v + t_{2} ⊗ v + t_{1} ⊗ v + v ⊗ t_{2} + v ⊗ t_{1} + v ⊗ s

Diagonal Valid!: Δ∂q == (1⊗∂ + ∂⊗1)Δq
All Diagonals Valid!
```

Computing Homology

- Computational Homology Project software: **CHomP**

Computing Homology

- Computational Homology Project software: **CHomP**
  - computes homology from incidence matrices

Computing Homology

- Computational Homology Project software: **CHomP**
  - computes homology from incidence matrices
  - can return generators

Computing Homology

- Computational Homology Project software: **CHomP**
    - computes homology from incidence matrices
    - can return generators
- we interface with CHomP to obtain H

Computing Homology

- Computational Homology Project software: **CHomP**
  - computes homology from incidence matrices
  - can return generators
- we interface with CHomP to obtain H
- use generators to construct cycle-selecting function g

Computing Δ_2 on UN

```
Mervins-MacBook-Pro:CellularChainParser mfansler$ python transfer.py data/unlinked.tex
H = H*(C) = { h0_0 = ['v'], h1_0 = ['a'], h1_1 = ['b'], h2_0 = ['t_{1}'] }

g =  a∂_{h1_0}
       + v∂_{h0_0}
       + t_{1}∂_{h2_0}
       + b∂_{h1_1}

Δg = (v ⊗ a + a ⊗ v)∂_{h1_0}
       + (v ⊗ v)∂_{h0_0}
       + (v ⊗ b + b ⊗ v)∂_{h1_1}
       + (v ⊗ t_{1} + t_{1} ⊗ v)∂_{h2_0}

Δ_2 = (h0_0 ⊗ h1_0 + h1_0 ⊗ h0_0)∂_{h1_0}
       + (h0_0 ⊗ h0_0)∂_{h0_0}
       + (h0_0 ⊗ h1_1 + h1_1 ⊗ h0_0)∂_{h1_1}
       + (h0_0 ⊗ h2_0 + h2_0 ⊗ h0_0)∂_{h2_0}
```

We (computationally) confirm that Δ_2 on the wedge of two pinched spheres is primitive.

Computing Δ_2 on LN

```
Mervins-MacBook-Pro:CellularChainParser mfansler$ python transfer.py data/linked.tex
H = H*(C) = { h0_0 = ['v'], h1_0 = ['a'], h1_1 = ['b'], h2_0 = ['t_{1}'] }

g =  a∂_{h1_0}
        + v∂_{h0_0}
        + t_{1}∂_{h2_0}
        + b∂_{h1_1}

Δg = (v ⊗ a + a ⊗ v)∂_{h1_0}
        + (v ⊗ v)∂_{h0_0}
        + (v ⊗ b + b ⊗ v)∂_{h1_1}
        + (b ⊗ a + a ⊗ b + v ⊗ t_{1} + t_{1} ⊗ v)∂_{h2_0}

Δ_2 = (h0_0 ⊗ h1_0 + h1_0 ⊗ h0_0)∂_{h1_0}
        + (h0_0 ⊗ h0_0)∂_{h0_0}
        + (h0_0 ⊗ h1_1 + h1_1 ⊗ h0_0)∂_{h1_1}
        + (h1_1 ⊗ h1_0 + h1_0 ⊗ h1_1 + h0_0 ⊗ h2_0 + h2_0 ⊗ h0_0)∂_{h2_0}
```

We (computationally) confirm that Δ_2 on the Hopf link is non-primitive!

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results
- add infrastructure to SageMath to do computations in there

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results
- add infrastructure to SageMath to do computations in there
    - supports chain complexes...

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results
- add infrastructure to SageMath to do computations in there
  - supports chain complexes...
  - but lacks DG-(co)algebra

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results
- add infrastructure to SageMath to do computations in there
  - supports chain complexes...
  - but lacks DG-(co)algebra
  - has polytopes for Associahedron and Multiplihedron...

Work in Progress

What is left to be done?

- finish implementing inductive step for Δ_n
- generate and validate diagonals of higher component Brunnian links
- compute some (hopefully interesting) results
- add infrastructure to SageMath to do computations in there
    - supports chain complexes...
    - but lacks DG-(co)algebra
    - has polytopes for Associahedron and Multiplihedron...
    - but lacks combinatorics and iterators

The End

Thank you!