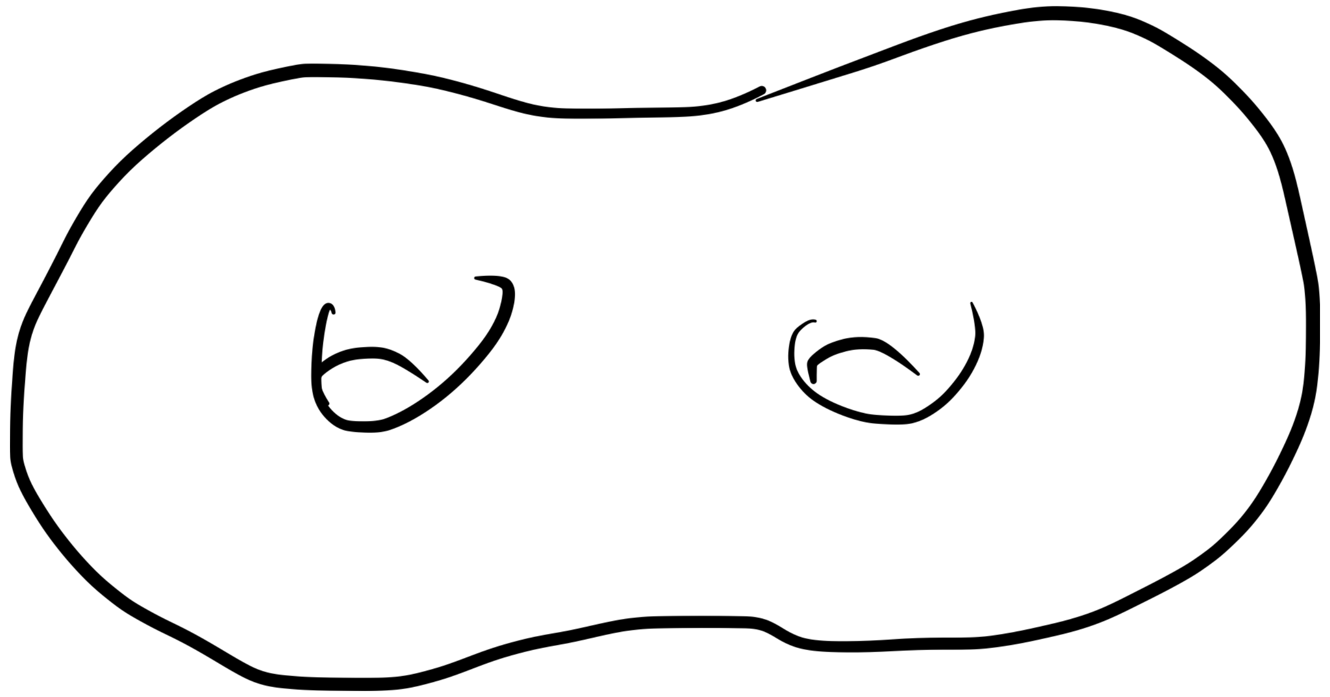
The word problem in the mapping class group is quasi-linear

Saul Schleimer
Mark Bell
ECM, 2024-07-16

Quasi-linear time

Suppose that S is a compact surface.

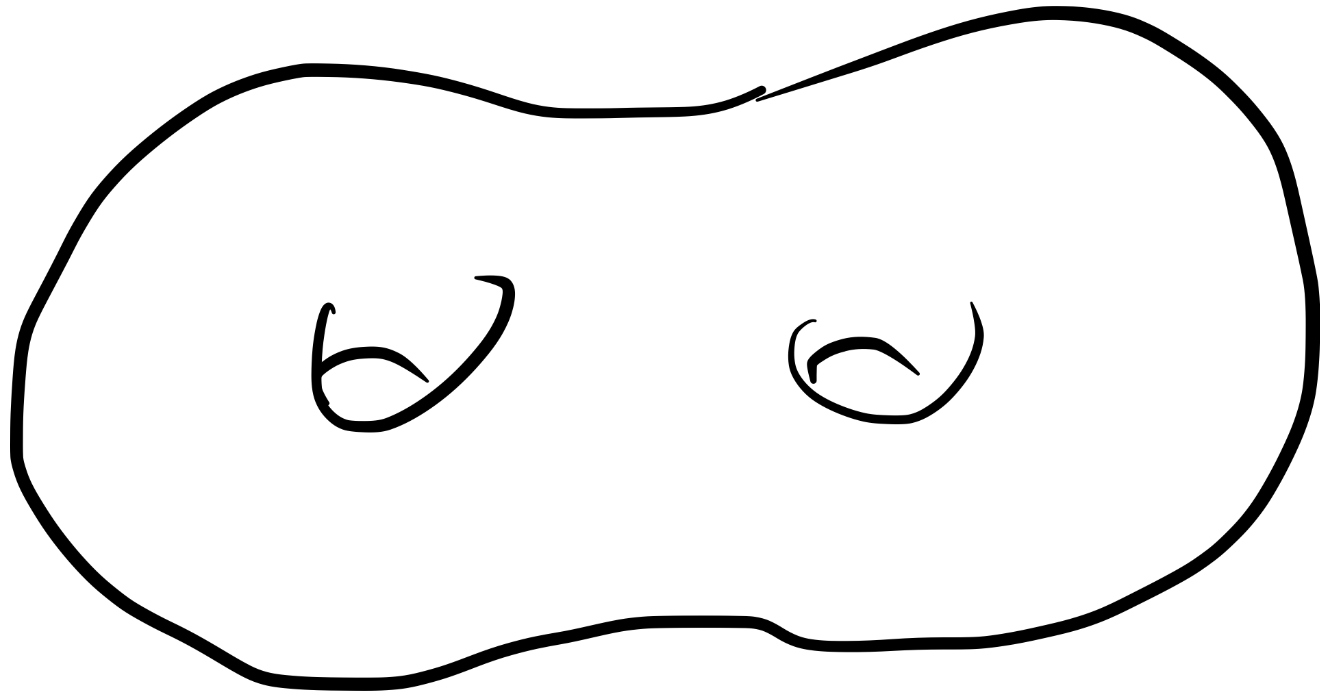


Let $\mathrm{MCG}(S)$ be the mapping class of S (equipped with a finite generating set).

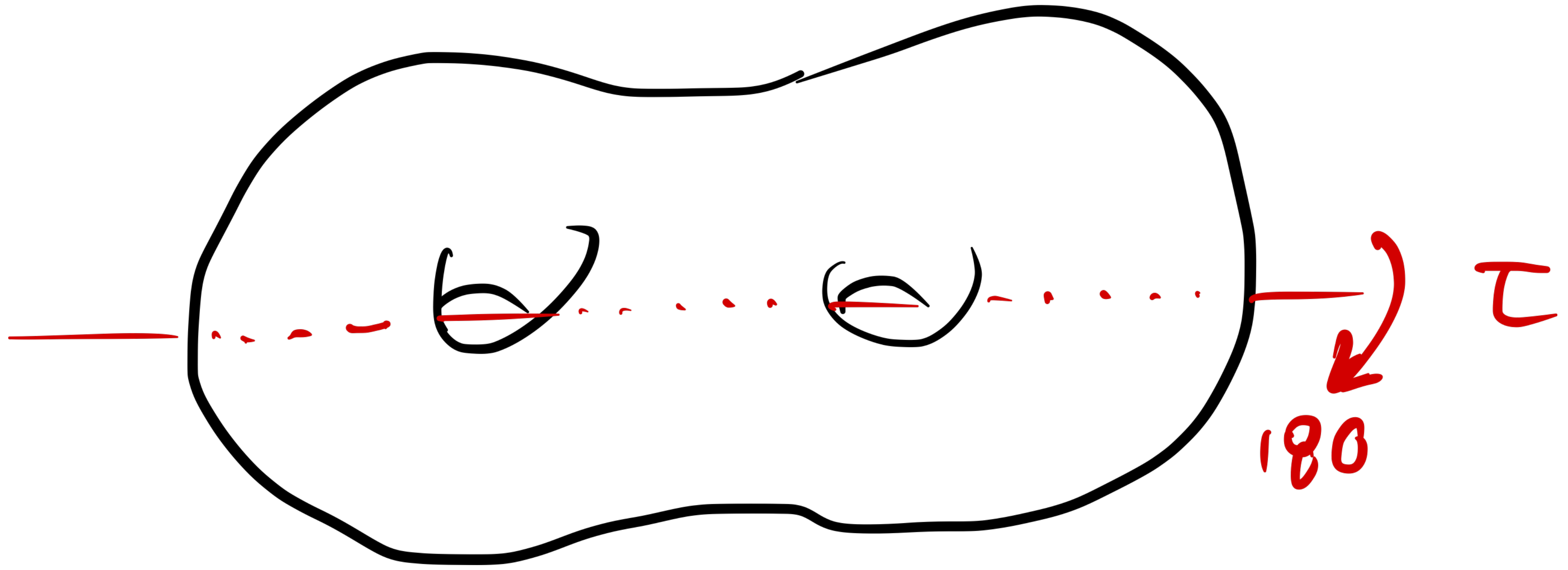
Theorem [Bell-Schleimer 2024]: There is a sub-quadratic time algorithm to solve the word problem in $\mathrm{MCG}(S)$.

The mapping class group

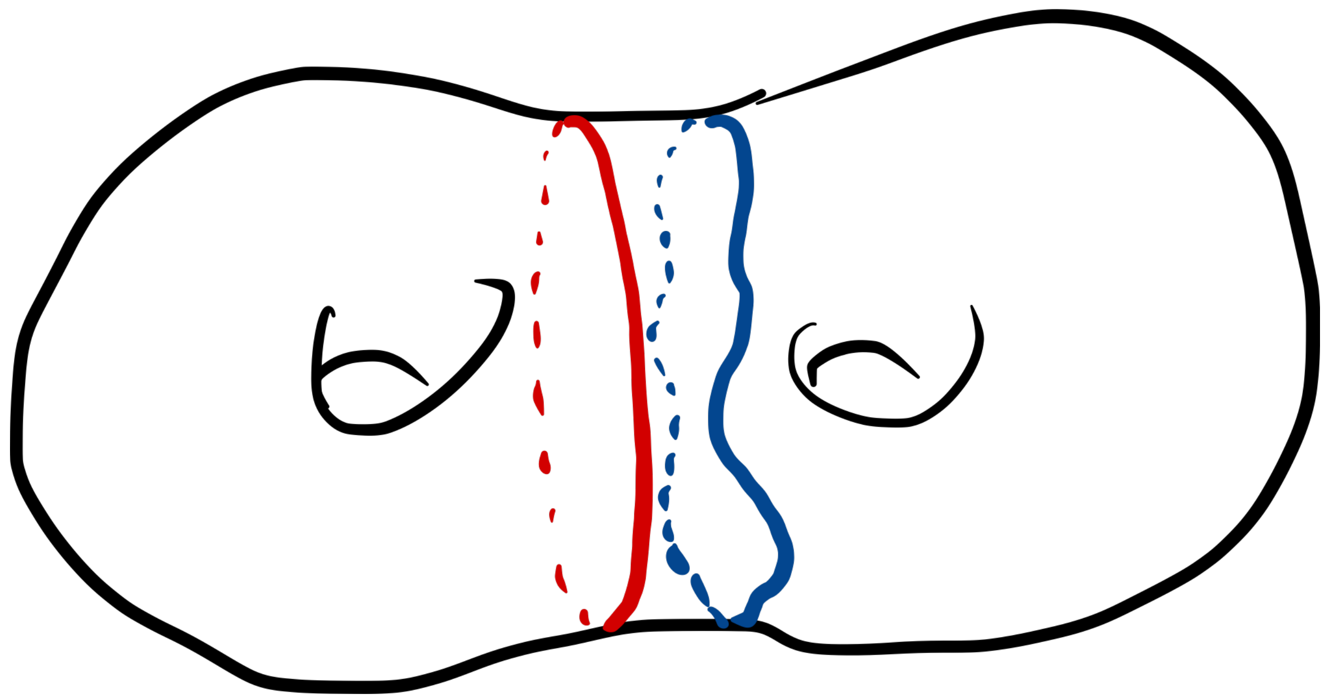
Suppose that S is a compact surface.



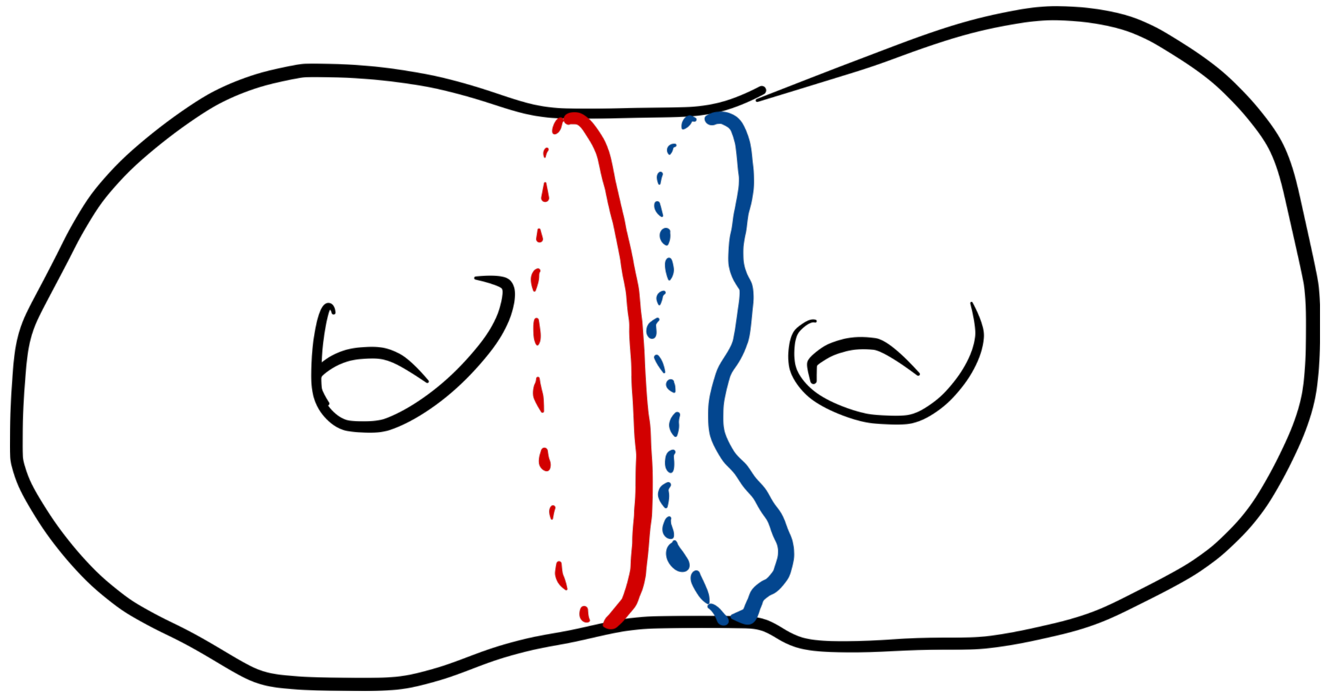
Suppose that $g, h \in \mathrm{Homeo}(S)$.



We write $g \simeq h$ if g and h are *isotopic.*



The mapping class group



We write $g \cong h$ if g and h are *isotopic.*

Dehn [1922] defines the *mapping class group* to be $\mathrm{MCG}(S) = \dfrac{\mathrm{Homeo}(S)}{\cong}$

Dehn [1922] also

- proves $\mathrm{MCG}(S)$ is finitely generated and
- gives two solutions for the *word problem* in $\mathrm{MCG}(S)$.

The word problem

A group G is *finitely generated* if there is a finite subset $X \subset G$ so that every element $g \in G$ can be realised as a finite product of elements from $X \cup X^{-1}$.

Example: $\mathbb{Z}^2$ is finitely generated by $\{x, y\}$, the standard basis vectors.

Example: $\mathbb{Q}$ is *not* finitely generated.

A finite list of elements from $X \cup X^{-1}$ is called a *word* over X. The length of the list is the *length* of the word. For example, $yxyx^{-1}y^{-1}y^{-1}$ has length six.

The word problem

Suppose that w is a word over X. The *word problem* [Dehn 1912] asks if the group element of G represented by w is the trivial element of G.

Example: the word $yxyx^{-1}y^{-1}y^{-1}$ represents the trivial element in $\mathbb{Z}^2$.

To solve the word problem, we need an *algorithm* that, given a word w over X, determines if $w =_G 1$.

The word problem

The word problem is the “first” problem in theory of finitely generated groups.

It is needed to build the Cayley graph (the first step in understanding the geometry of a group).

As an example, we can solve the word problem in $\mathbb{Z}^2$ by maintaining a pair of stacks (one for each generator).

The word problem

We measure the *time complexity* of our algorithm in terms of the length n of the given word w.

Example: the pair-of-stacks algorithm for the word problem in $\mathbb{Z}^2$ takes time $O(n)$ — that is, linear in n with constants depending only on G and X.

Exercise: Fix $d > 2$ and generate $\mathrm{SL}(d, \mathbb{Z})$ by elementary matrices. Now solve the word problem. Now bound the time complexity of your algorithm.

The mapping class group

Dehn [1922] gives two solutions to the word problem in $\mathrm{MCG}(S)$.

Solution (A), via the "action" of $\mathrm{MCG}(S)$ on $\pi_1(S)$, has time complexity $2^{O(n)}$.

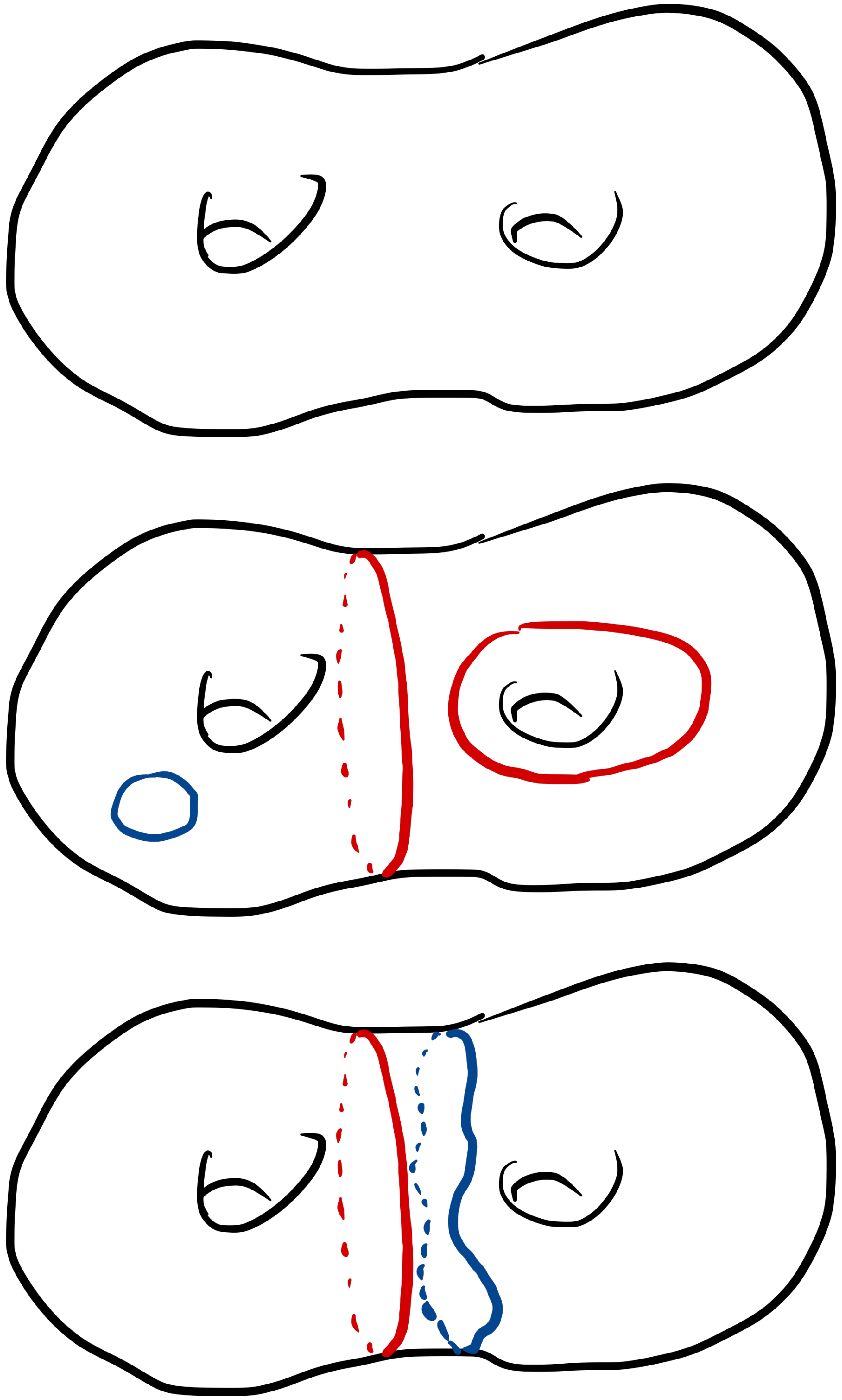
Solution (B), via the action of $\mathrm{MCG}(S)$ on $\mathscr{C}(S)$, has time complexity $O(n^2)$.

Multi-curves

Suppose that S is a compact surface.

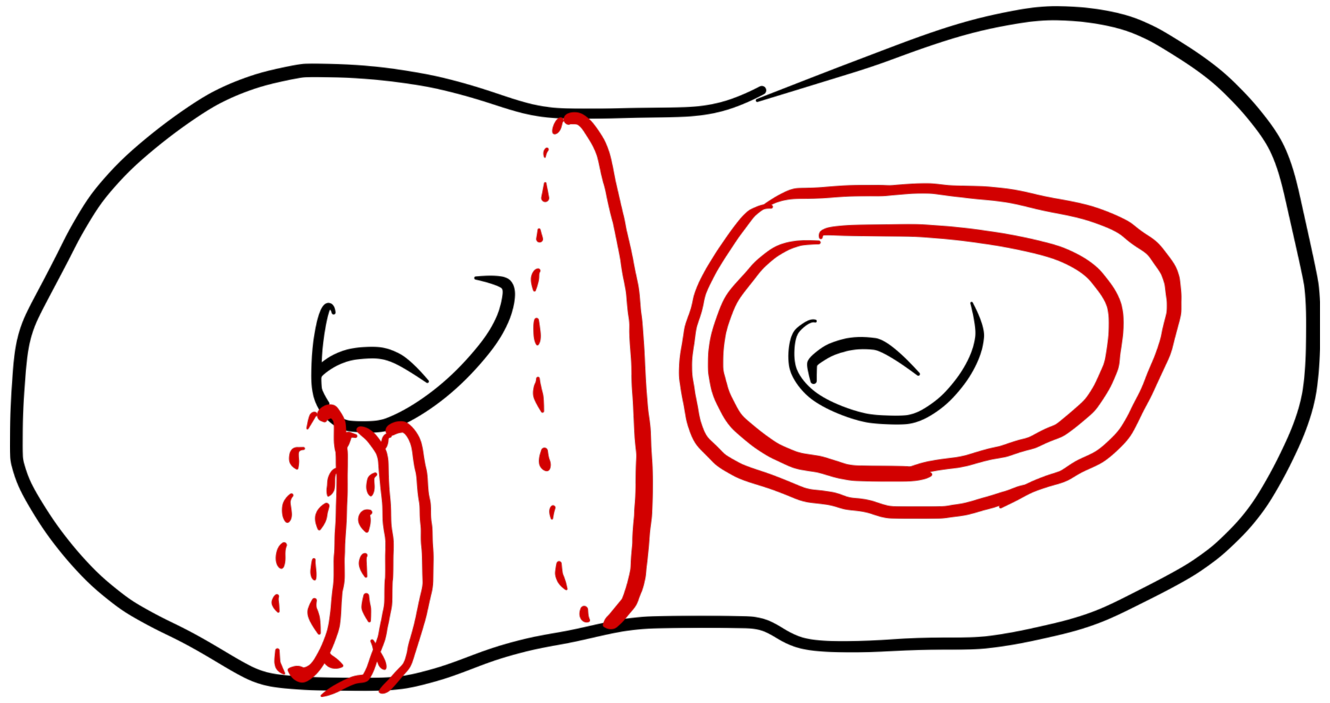
Suppose that α and β are curves in S.

We write $\alpha \cong \beta$ if α and β are *isotopic.*

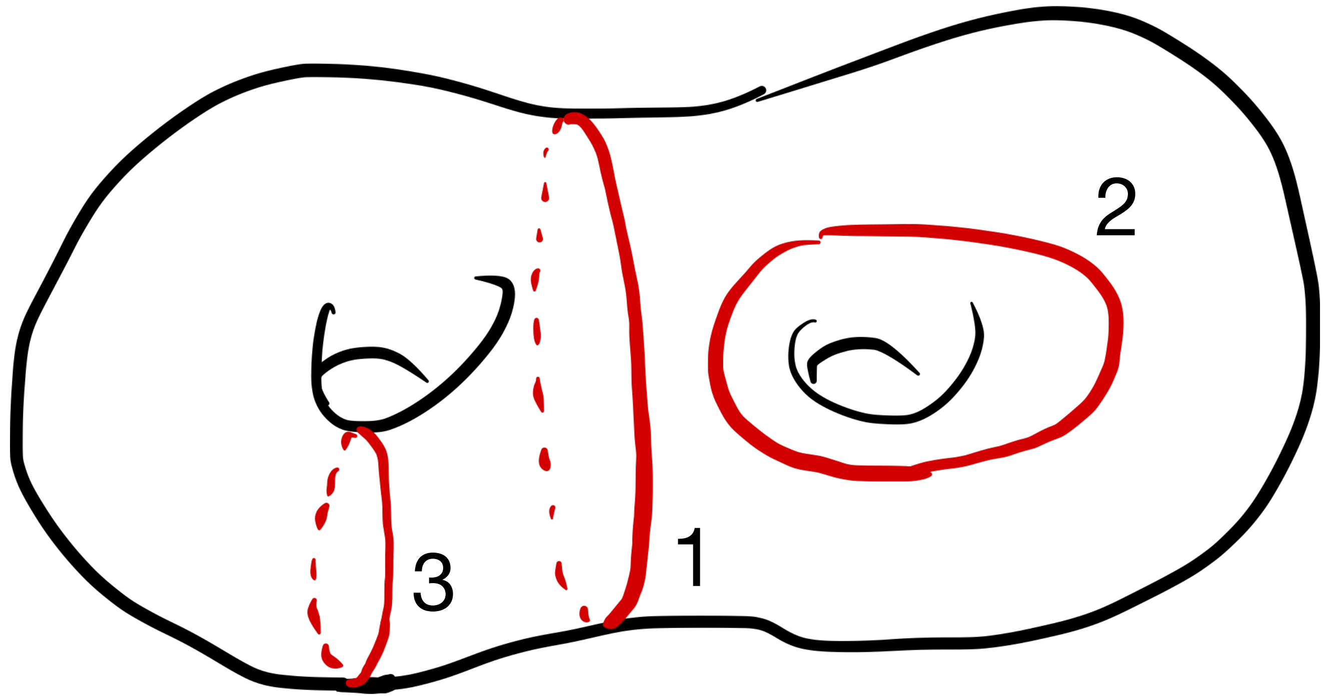


Multi-curves (with weights)

A *multi-curve* in S is a finite disjoint union of curves.



We can simplify the figures by using *weights*.



We define $\mathscr{C}(S)$ to be the set of multi-curves in S, considered up to isotopy.

The mapping class group

(A) via action on $\pi_1(S)$ has time complexity $2^{O(n)}$.

S- [2008] accelerates to poly-time using *straight-line programs.*

(B) via action on $\mathscr{C}(S)$ has time complexity $O(n^2)$. Other quadratic time algorithms include the following.

Penner [1982] implements Thurston's action of $\mathrm{MCG}(S)$ on $\mathrm{PML}(S)$

Mosher [1995] gives an automatic structure on $\mathrm{MCG}(S)$ for $\partial S \neq \varnothing$

Takarajima [1999] gives an automatic structure on $\mathrm{MCG}(S)$ for $\partial S = \varnothing$

Hamidi-Tehrani [2000] gives an action on $\mathrm{PML}(S)$ using Birman-Series $\pi_1(S)$-tracks

D.Thurston [2008] computes the geometric intersection number using smoothing lemma

Dynnikov [2022] computes the geometric intersection number using *curve shortening*

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n\ldots$

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n\log(n)\ldots$

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n \log(n)\ \log(n)\ldots$

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n \log(n)\ \log(n)\ \log(n)\ldots$

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n \log(n)\ \log(n)\ \log(n))$.

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n \log^3(n))$.

Quasi-linear time

Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(n \log^3(n))$.

Theorem [Bell-Schleimer 2024]: There is an algorithm that, given *weighted standard tracks* carrying *multi-curves* α and β, computes the *geometric intersection number* $\iota(\alpha, \beta)$ in time $O(n \log^2(n))$.

Theorem [Bell-Schleimer 2024]: There is an algorithm that, given a *weighted train track* carrying a *multi-curve* α, performs *curve shortening* in time $O(n \log^2(n))$.

Quasi-linear time

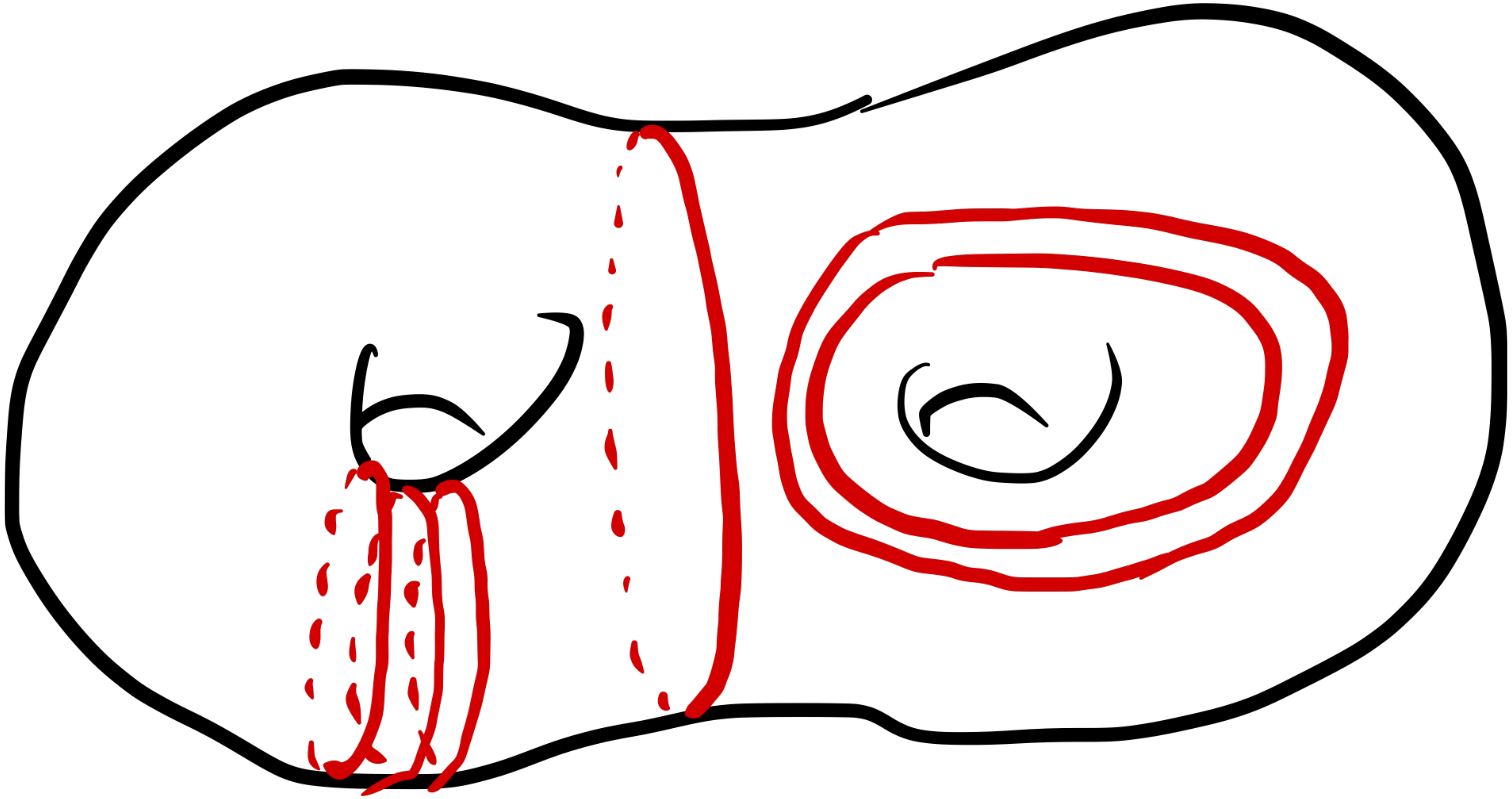
Theorem [Bell-Schleimer 2024]: There is an algorithm to solve the word problem in $\mathrm{MCG}(S)$ in time $O(M(n)\log^2(n))$.

Theorem [Bell-Schleimer 2024]: There is an algorithm that, given *weighted standard tracks* carrying *multi-curves* α and β, computes the *geometric intersection number* $\iota(\alpha, \beta)$ in time $O(M(n)\log(n))$.

Theorem [Bell-Schleimer 2024]: There is an algorithm that, given a *weighted train track* carrying a *multi-curve* α, performs *curve shortening* in time $O(M(n)\log(n))$.

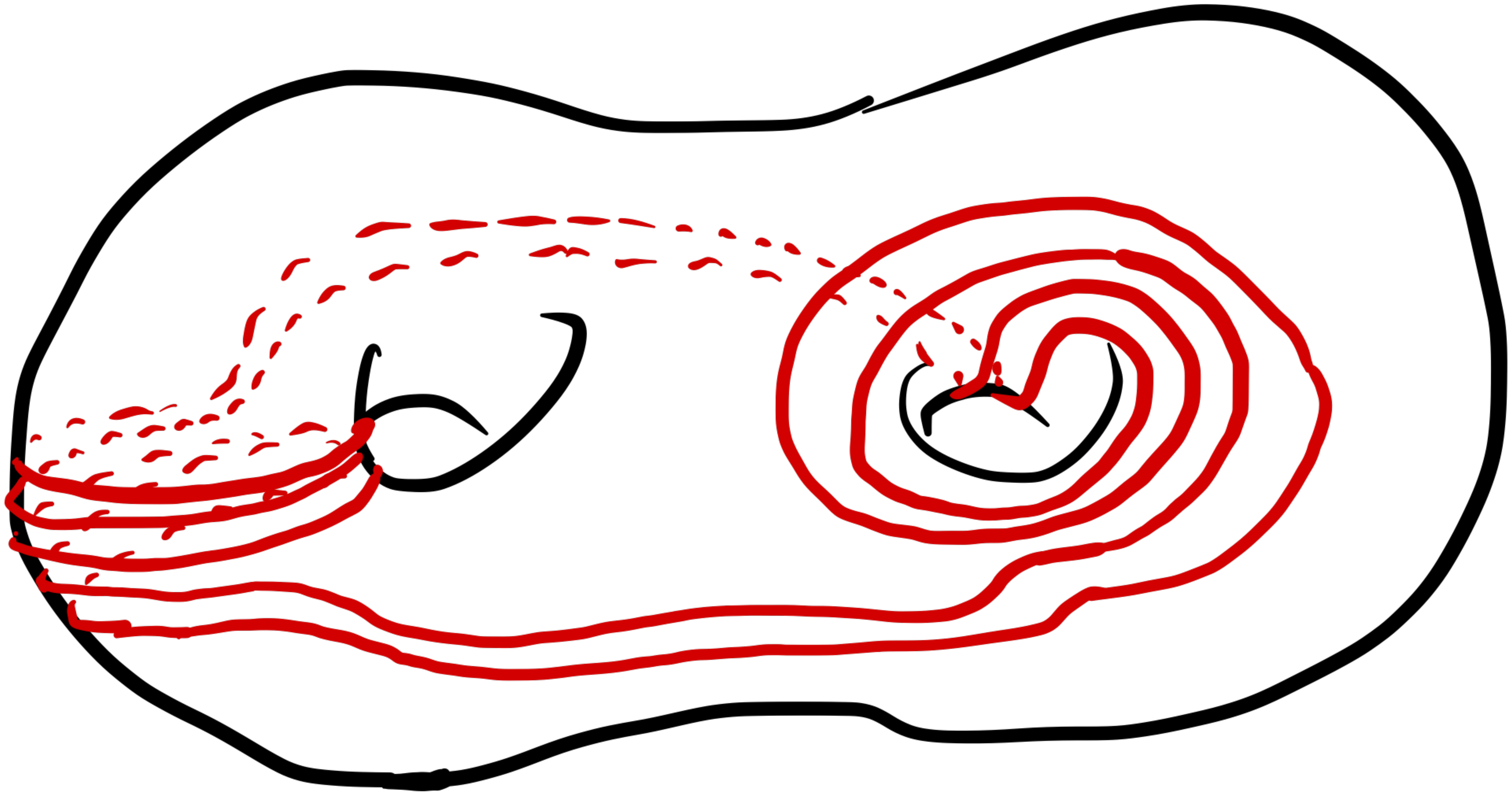
Weighted train tracks

Here is a multi-curve. It has six components.



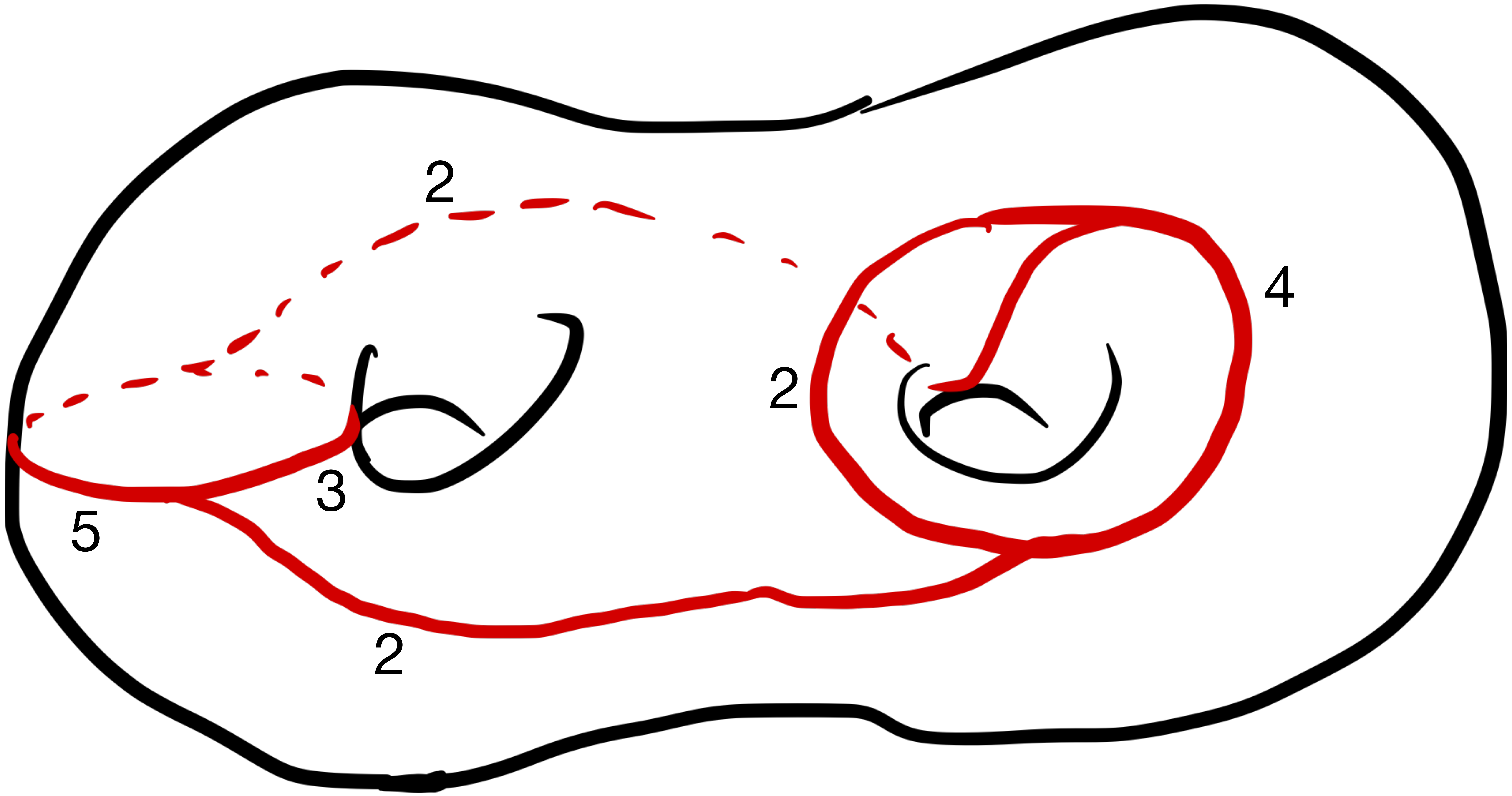
Weighted train tracks

A more complicated multi-curve. **Exercise:** Count the components!



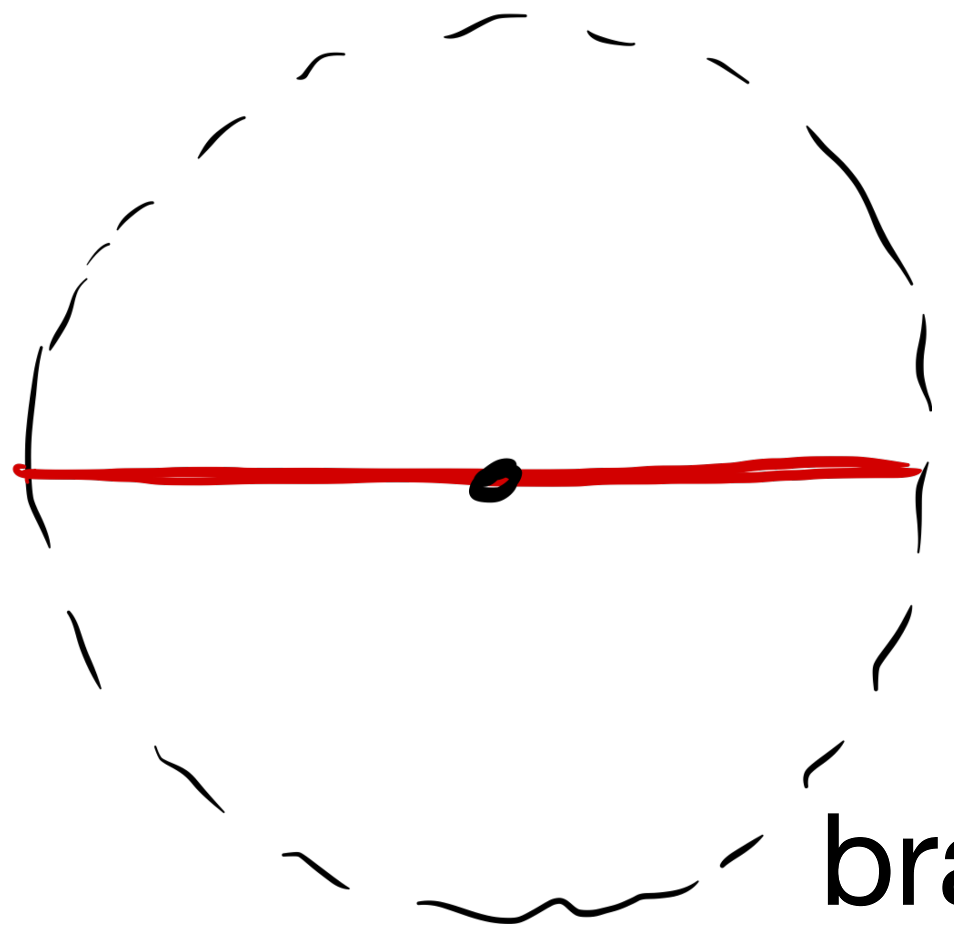
Weighted train tracks

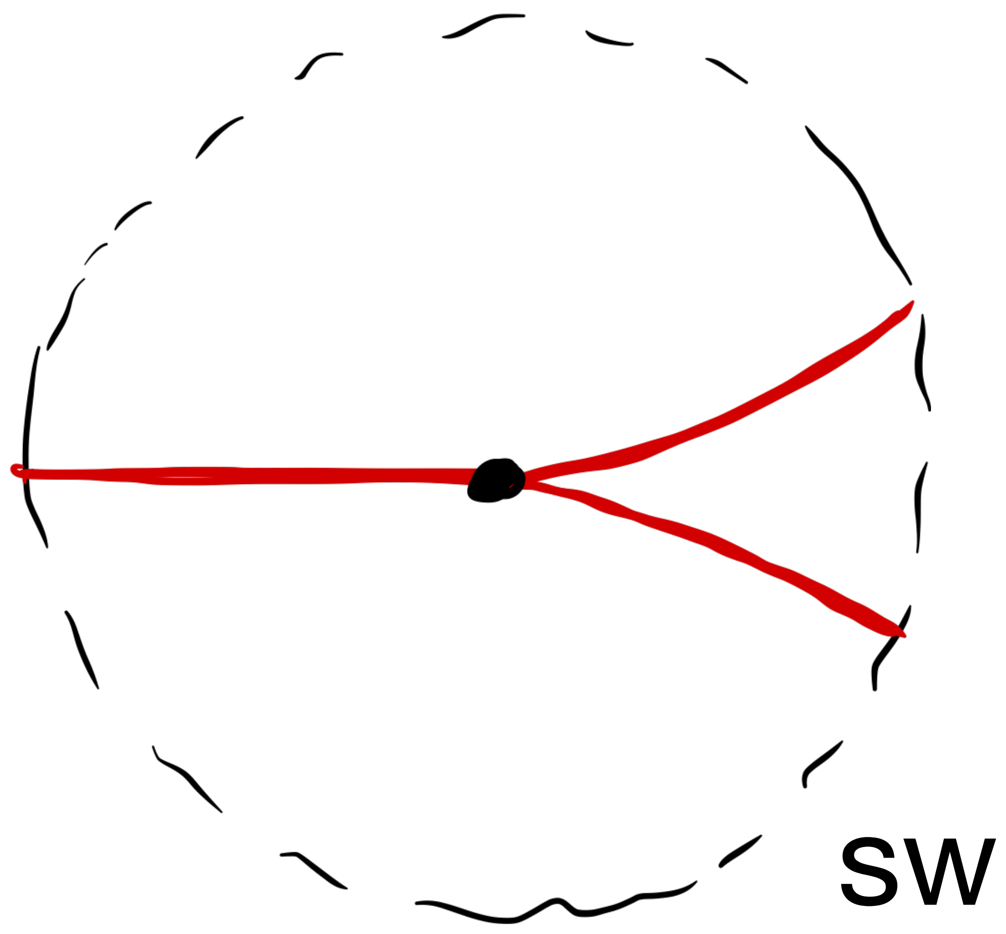
We can represent complicated multi-curves using *weighted train tracks.*



Weighted train tracks

A *train track* $\tau \subset S$ is a closed subset with the following local models.

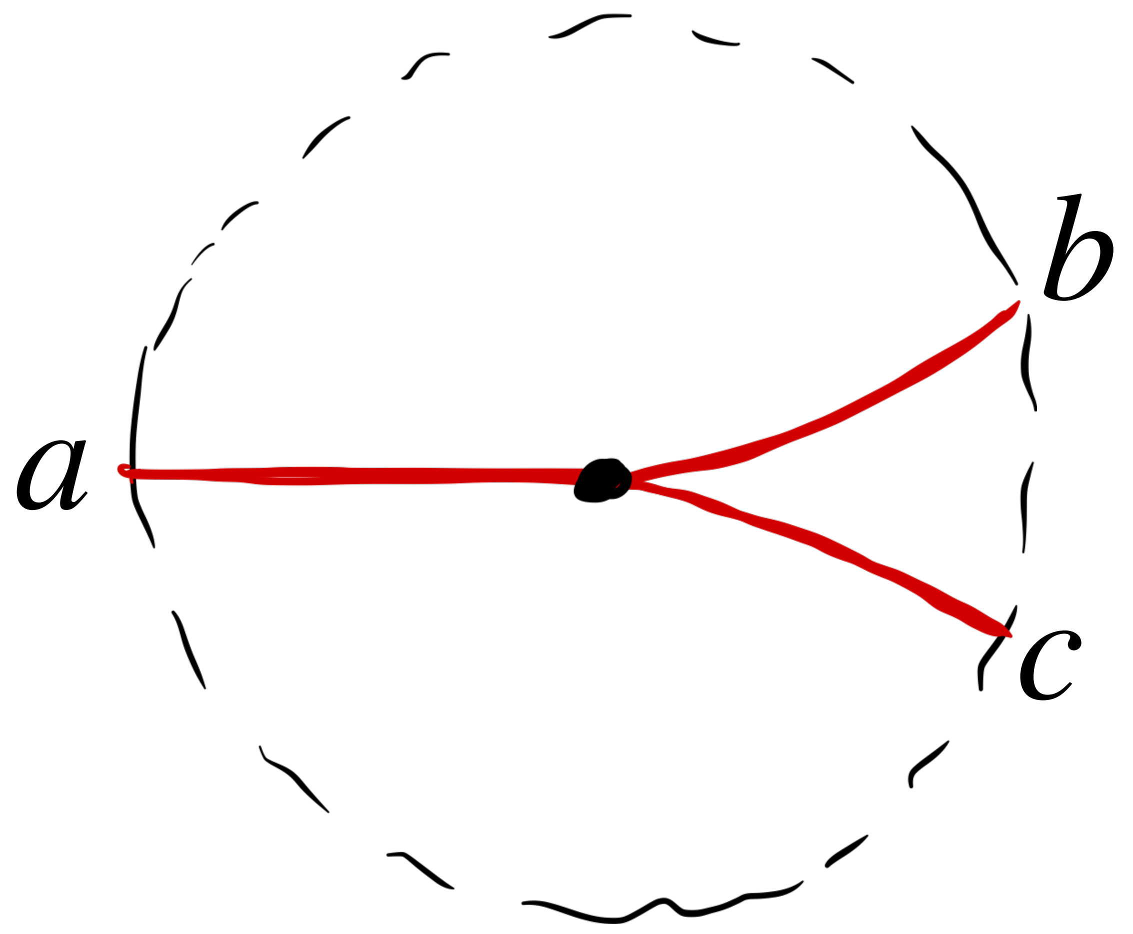

branch point

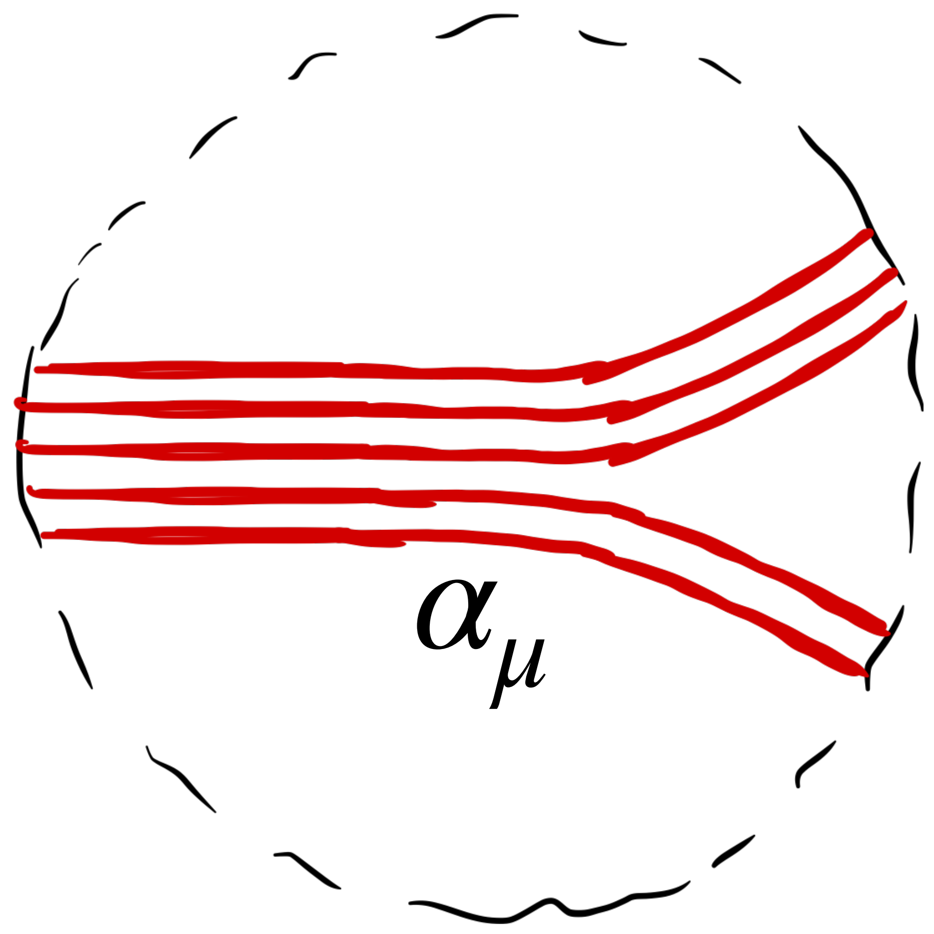

switch

We define $S(\tau)$ to be the set of *switches* in τ. We define $B(\tau)$ to be the set of *branches* in τ: that is, the connected components of $S - S(\tau)$.

Weighted train tracks

A *weighting* $\mu : B(\tau) \to \mathbb{N}$ is any function satisfying the *switch equalities*.





That is, for each switch $s \in S(\tau)$ we have $\mu(a) = \mu(b) + \mu(c)$.

By taking parallel strands we can build a multi curve $\alpha_\mu \in \mathscr{C}(S)$.

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Count the number of components of α.

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Count the number of components of α.

Decide if there is a mapping class $f \in \mathrm{MCG}(S)$ so that $f(\alpha) = \beta$.

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Count the number of components of α.

Decide if there is a mapping class $f \in \mathrm{MCG}(S)$ so that $f(\alpha) = \beta$.

Compute $[\alpha] \in H_1(S, \mathbb{Z})$.

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Count the number of components of α.

Decide if there is a mapping class $f \in \mathrm{MCG}(S)$ so that $f(\alpha) = \beta$.

Compute $[\alpha] \in H_1(S, \mathbb{Z})$.

Compute $[\alpha] \cdot [\beta]$ (algebraic intersection number).

Weighted train tracks

Suppose that τ is a train track. Suppose that μ and ν are given weightings on τ. Suppose that $\alpha = \alpha_\mu$ and $\beta = \alpha_\nu$ are the resulting multi-curves.

There are various questions we can ask:

Count the number of components of α.

Decide if there is a mapping class $f \in \mathrm{MCG}(S)$ so that $f(\alpha) = \beta$.

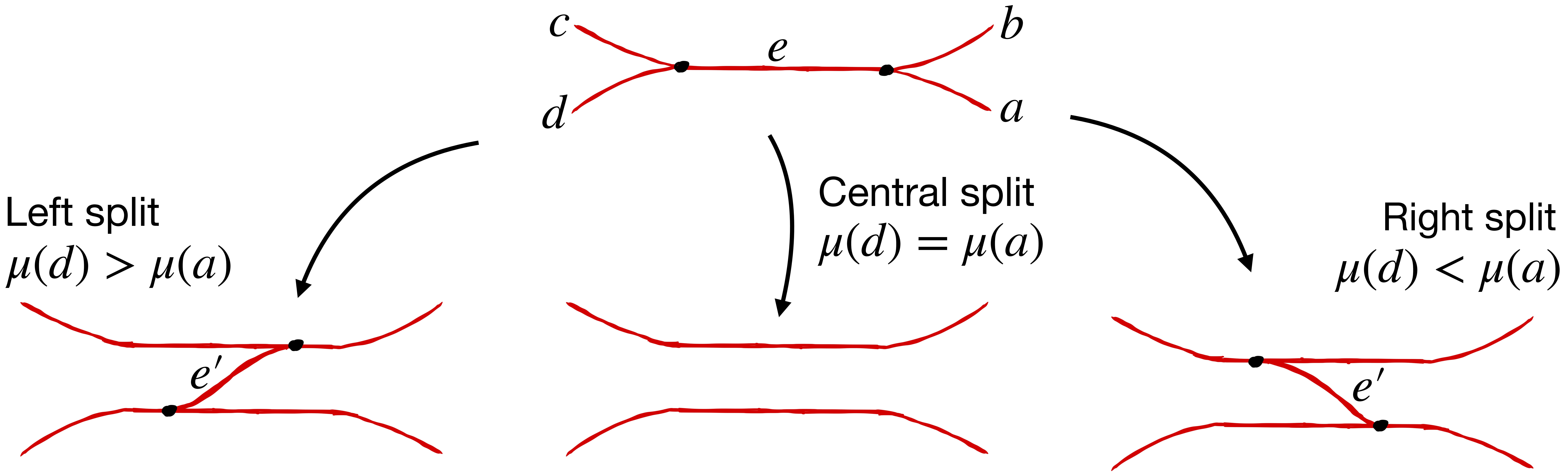
Compute $[\alpha] \in H_1(S, \mathbb{Z})$.

Compute $[\alpha] \cdot [\beta]$ (algebraic intersection number).

Compute $\iota(\alpha, \beta)$ (geometric intersection number).

Curve shortening

Suppose that (τ, μ) is a track and weighting. We may *split* τ according to μ to obtain a new track τ' equipped with the induced weighting μ'.

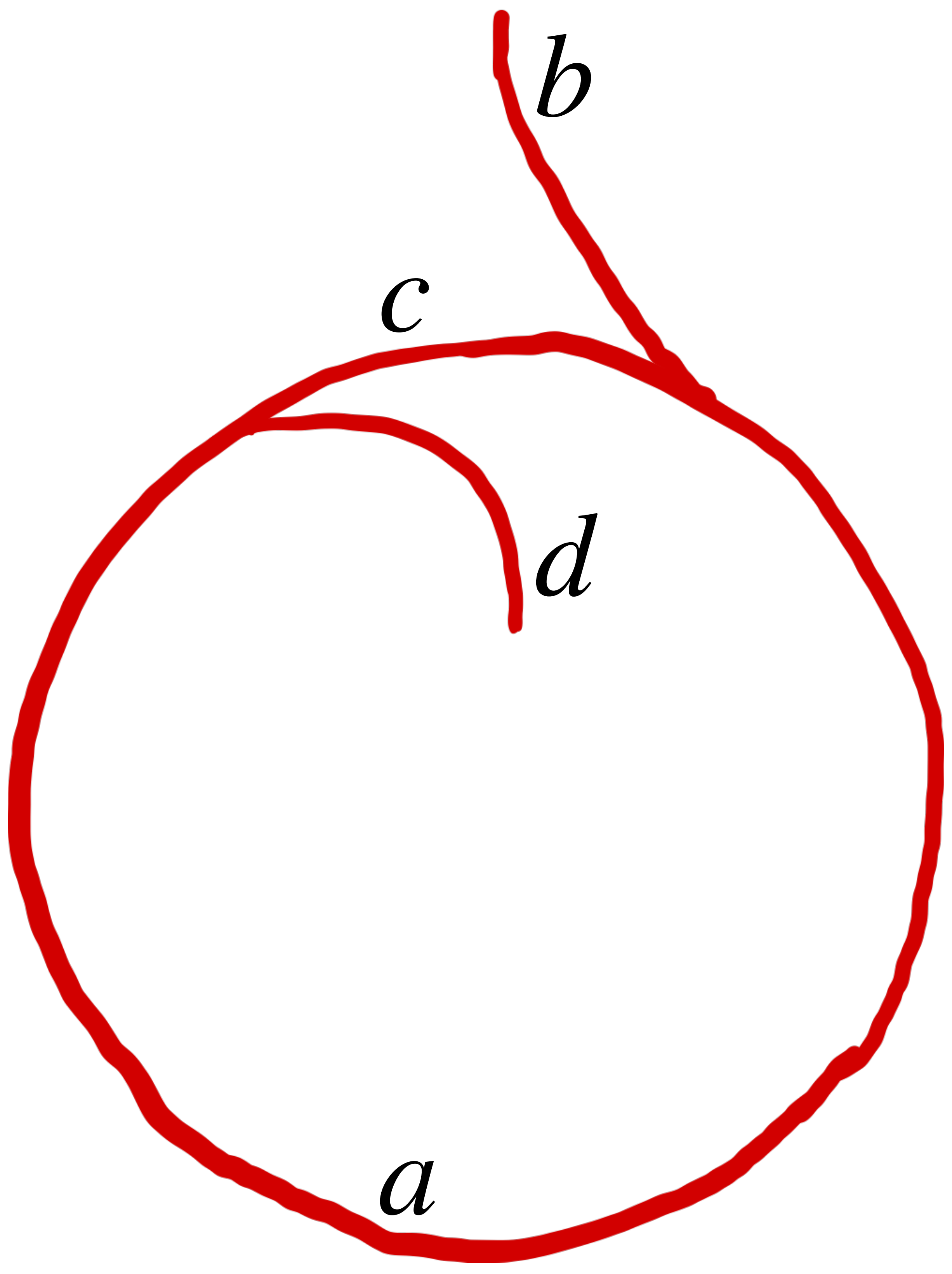


$\mu'(e') = \mu(e) - \mu(a) - \mu(c)$

$\mu'(e') = \mu(e) - \mu(b) - \mu(d)$

Curve shortening

Suppose that (τ, μ) is a train track with weights. Suppose that $\gamma \subset \tau$ is a *combed train loop*. Then we may untwist τ according to μ, say k times, to obtain the same track τ equipped with the induced weighting μ'.



$$\mu'(a) = \mu(a) - k \cdot \mu(b)$$

$$\mu'(c) = \mu(c) - k \cdot \mu(b)$$

Curve shortening versus euclidean algorithm

Curve shortening	Euclidean algorithm
(τ, μ)	$(u, v) \in \mathbb{N}^2$
split	subtraction
untwist	division with remainder
# of components of α_μ	$\gcd(a, b)$
$\mathrm{MCG}(S)$	$\mathrm{GL}(2,\mathbb{Z})$

Curve shortening

Theorem: There is a constant $k = k(S)$ with the following property. Suppose that (τ, μ) is a train track with weights. Then there is a splitting and untwisting sequence (τ_i, μ_i) starting at (τ, μ), ending at a track without switches, and with the bit-size of μ_{i+k} at least one less than that of μ_i.

This (modulo subtle details) gives us an $O(n^2)$ algorithm.

This version of curve shortening, and the usual euclidean algorithm, are both $O(n^2)$ for essentially the same reasons.

Half-GCD algorithm

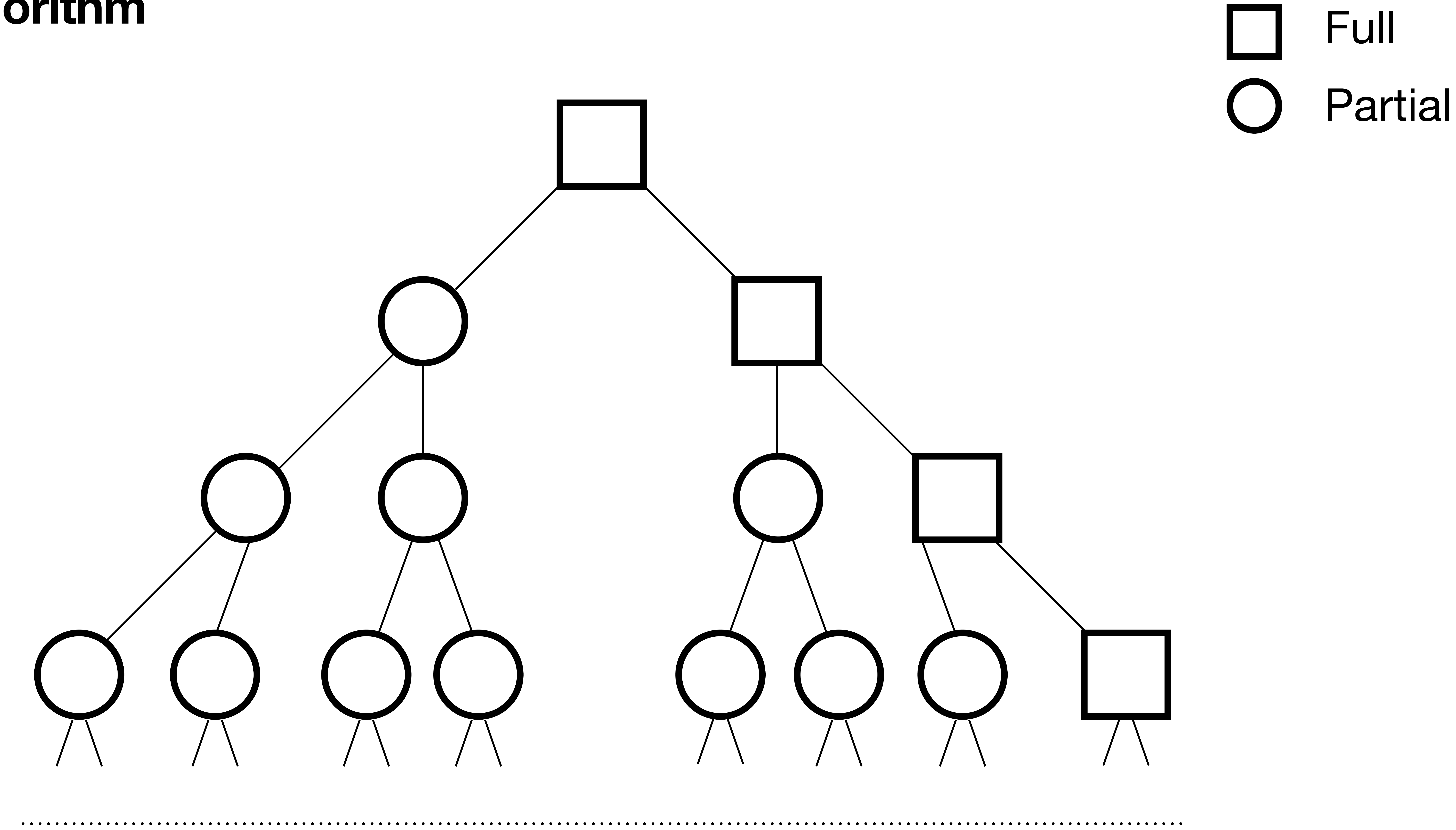
```
a = 8345399854518752, b = 5743132135431331  # full
A = 83453998        , B = 57431321          # partial

cf(a,b)=[1,2,4,1,4,1,14,1,11,1,1,1,3,1,3,4,1,11,1,6,1,5, …
cf(A,B)=[1,2,4,1,4,1,14,1,11,1,1,3,1,13,1,1,1,2,4]
```

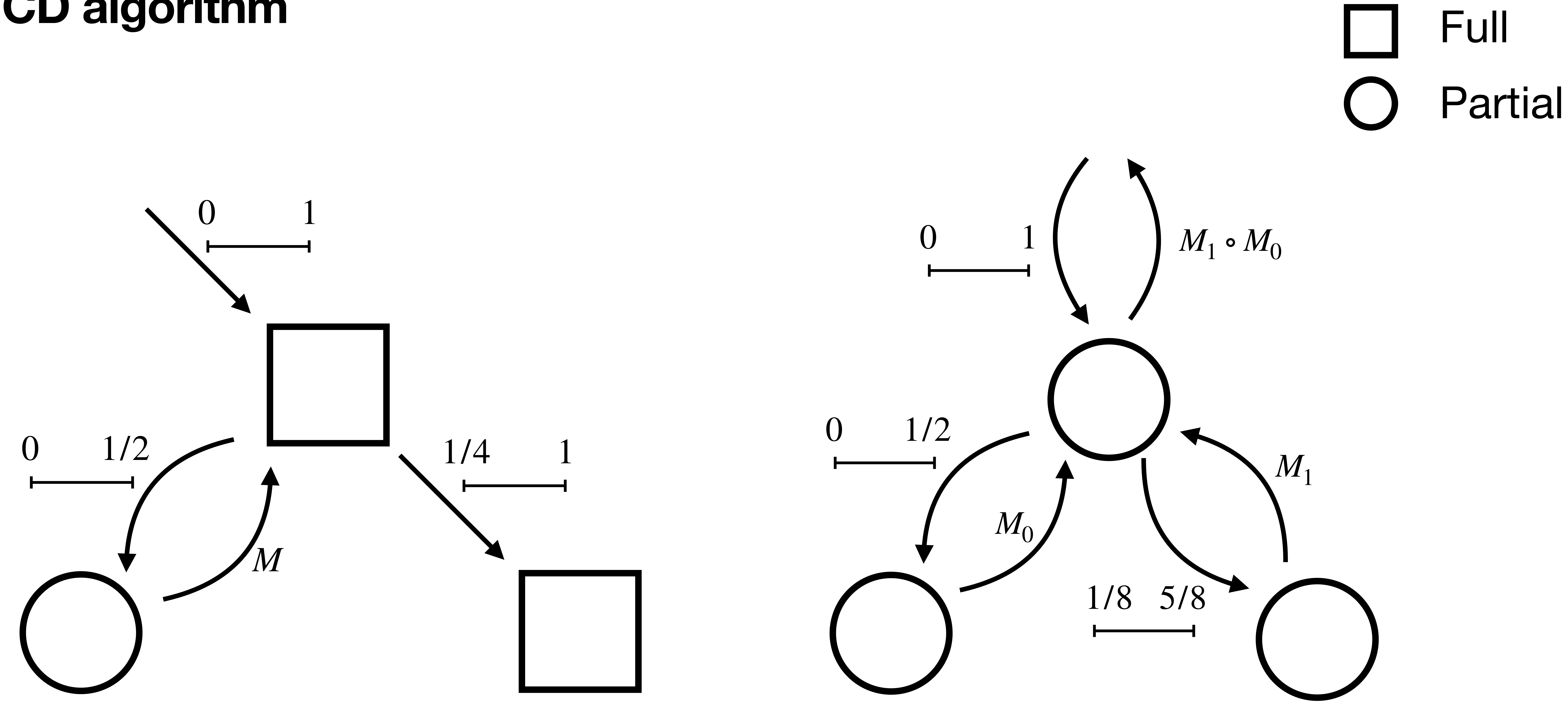
That is, the continued fractions of (a, b) and of (A, B) have a common prefix.

This leads to a recursive algorithm, called the *half-GCD*, which computes continued fraction expansions in time $O(M(n)\log(n))$.

Half-GCD algorithm



Half-GCD algorithm



Accelerated curve shortening

(Again, ignore untwisting in order to simplify the discussion.)

We only need the “most significant bits” of $\mu\colon B(\tau) \to \mathbb{N}$ to determine the first split. Similarly, we only need $O(\ell)$ significant bits of μ in order to determine the first ℓ splits.

This idea leads to a recursive curve shortening algorithm, modelled on the half-GCD, which finds the splitting sequence τ_i — the weights μ_i are only needed to full precision along the rightmost branch of the call tree.

Thank you!

Saul Schleimer
Mark Bell
ECM, 2024-07-16