Fast Nielsen-Thurston Classification

Balázs Strenner

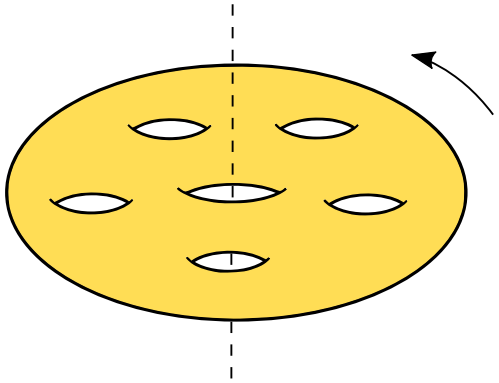
Georgia Institute of Technology

joint with Dan Margalit, S. Öykü Yurttaş

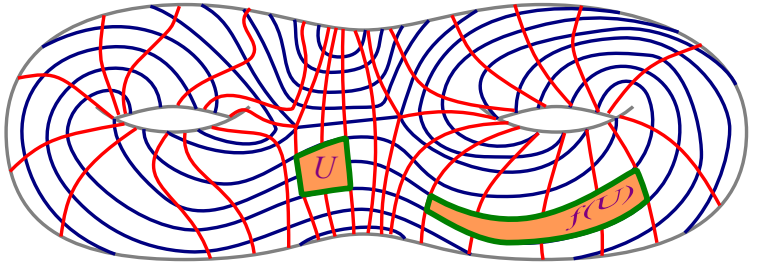
Geometry of Teichmüller space and mapping class groups
University of Warwick
April 9-13, 2018

The Nielsen-Thurston Classification

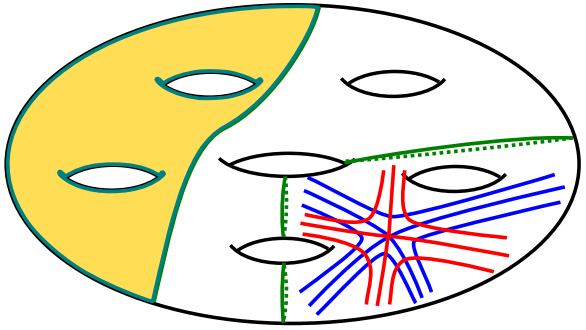
Mod(S)



Finite order



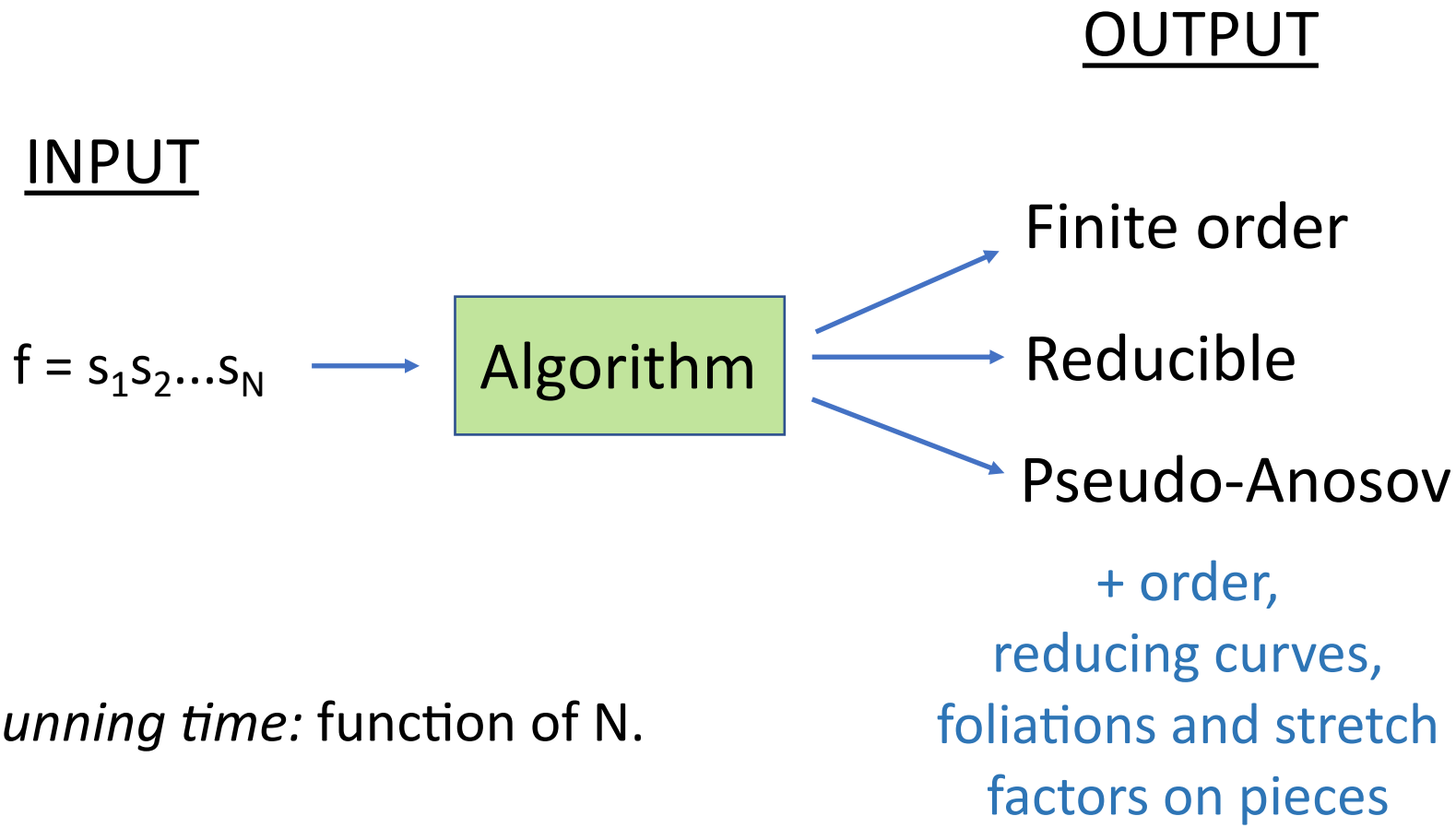
Pseudo-Anosov



Reducible

The Nielsen-Thurston Classification Problem

Fix finite generating set for Mod(S).



Running time: function of N.

Prior results on Classification

Higher genus surfaces, exponential algorithm: Thurston & Mosher ‘70s-’82, Bestvina-Handel ’95, Hamidi-Tehrani&Chen’96, Koberda-Mangahas ’13, Bell: NP,co-NP (arXiv ‘16)

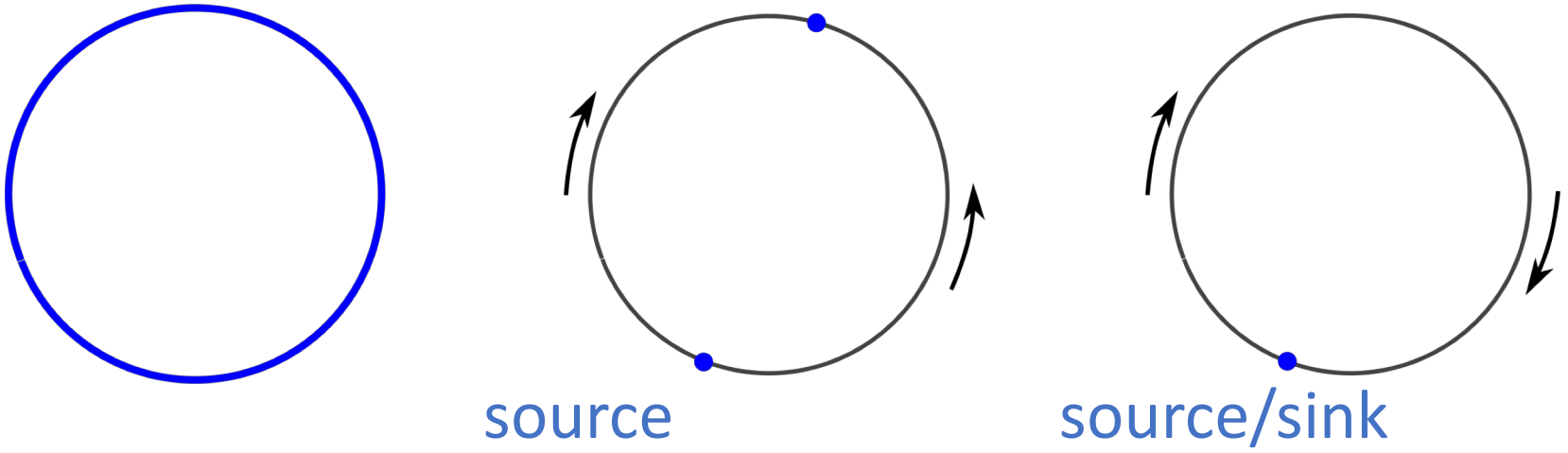
Braids, quadratic algorithm: Los ’93, Bernadete-Gutierrez-Nitecki ’95, Calvez ’13

Main Theorem

Theorem (Margalit-S-Yurttaş): Quadratic-time algorithm for the Nielsen-Thurston Classification Problem.

Bell-Webb (arXiv '16): Polynomial-time algorithm to determine Nielsen-Thurston type and find: order, reducing curves, and translation length in the curve complex.

Isometries of $\mathbb{H}^2$

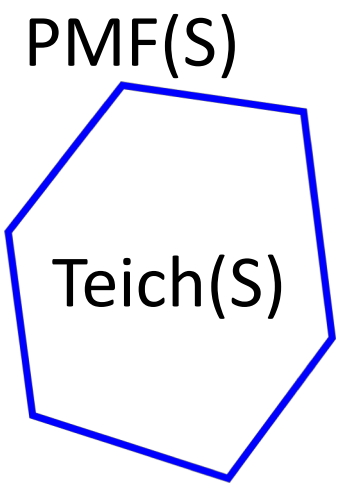


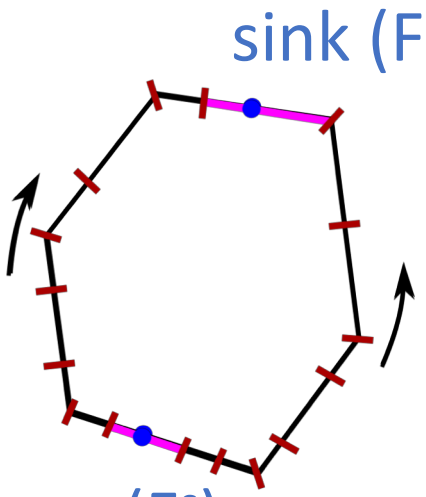
elliptic
(up to power)

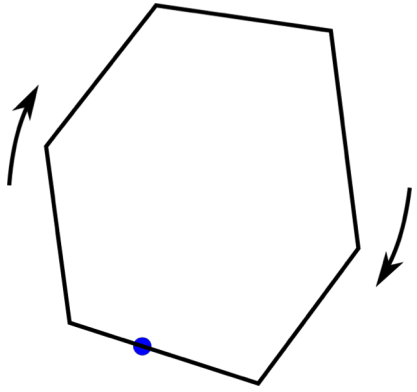
hyperbolic

parabolic

The mapping class group







finite order
(up to power)

pseudo-Anosov

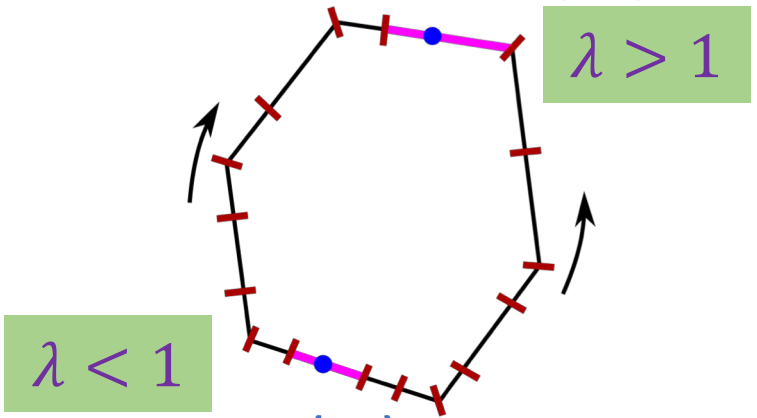
Reducible

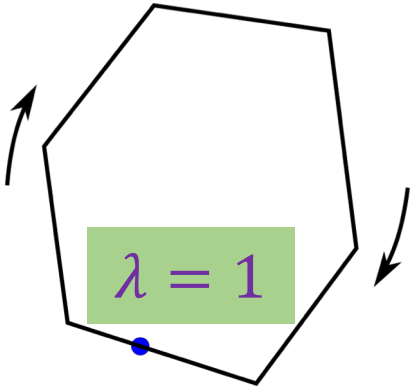
Thurston (70s): Compute the piecewise linear map and find all the eigenvectors.

Exponentially many pieces

PMF(S)

Teich(S)





source/sink
(invariant curves)

finite order
(up to power)

pseudo-Anosov

Reducible

Iterate!

~2010:

Toby Hall: `Dynn`, first iterate, then compute eigenvectors.

😀

Unknown rate of convergence
Unknown behavior in reducible case

😩

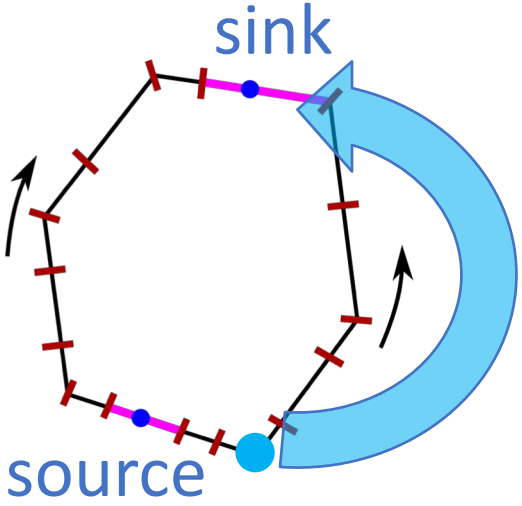
Bit later:

Margalit-S-Yurttaş: Let's try to prove that convergence is **fast**.

✓

Bell-Schleimer: Let's try to prove that convergence is **slow**.

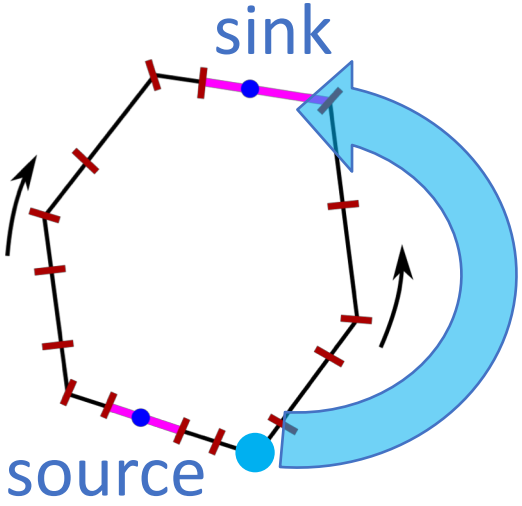
✓



Iterate!

Margalit-S-Yurttaş: Let's try to prove that convergence is **fast**.

Bell-Schleimer: Let's try to prove that convergence is **slow**.



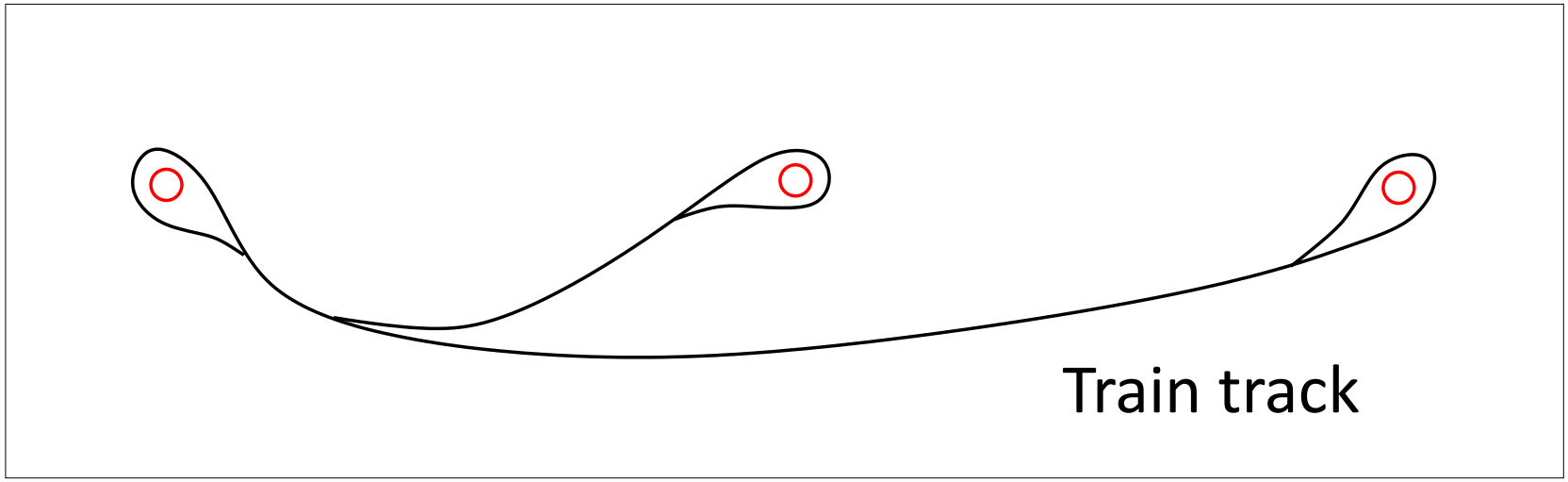
Bell-Schleimer ('15): convergence can be exponentially slow in the linear piece.

Margalit-S-Yurttaş ('15): We get into the attracting linear piece in O(N) iterations.

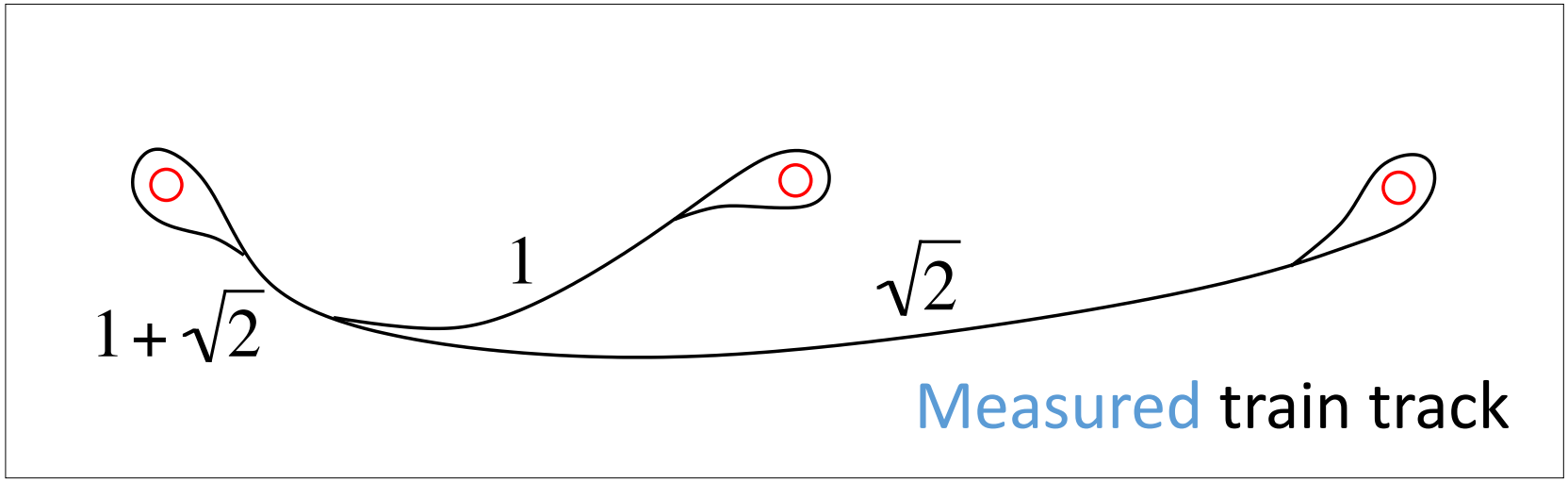
Algorithm v1

Linear charts on MF = train tracks

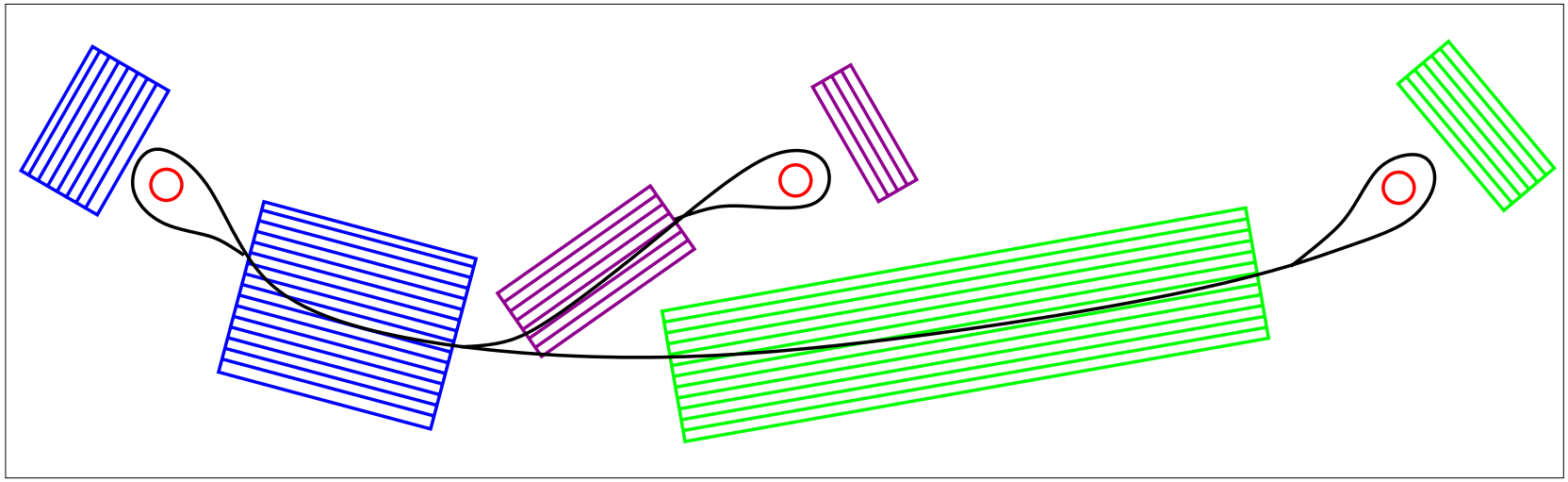
MF = measured foliations



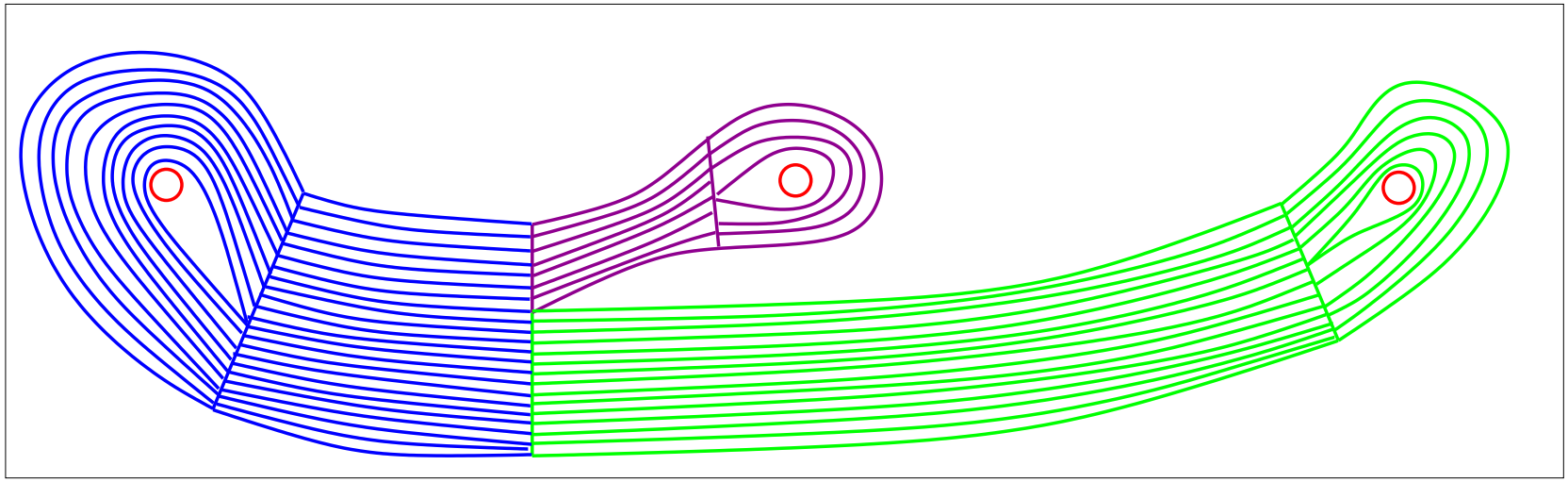
Linear charts on MF = train tracks



Linear charts on MF = train tracks



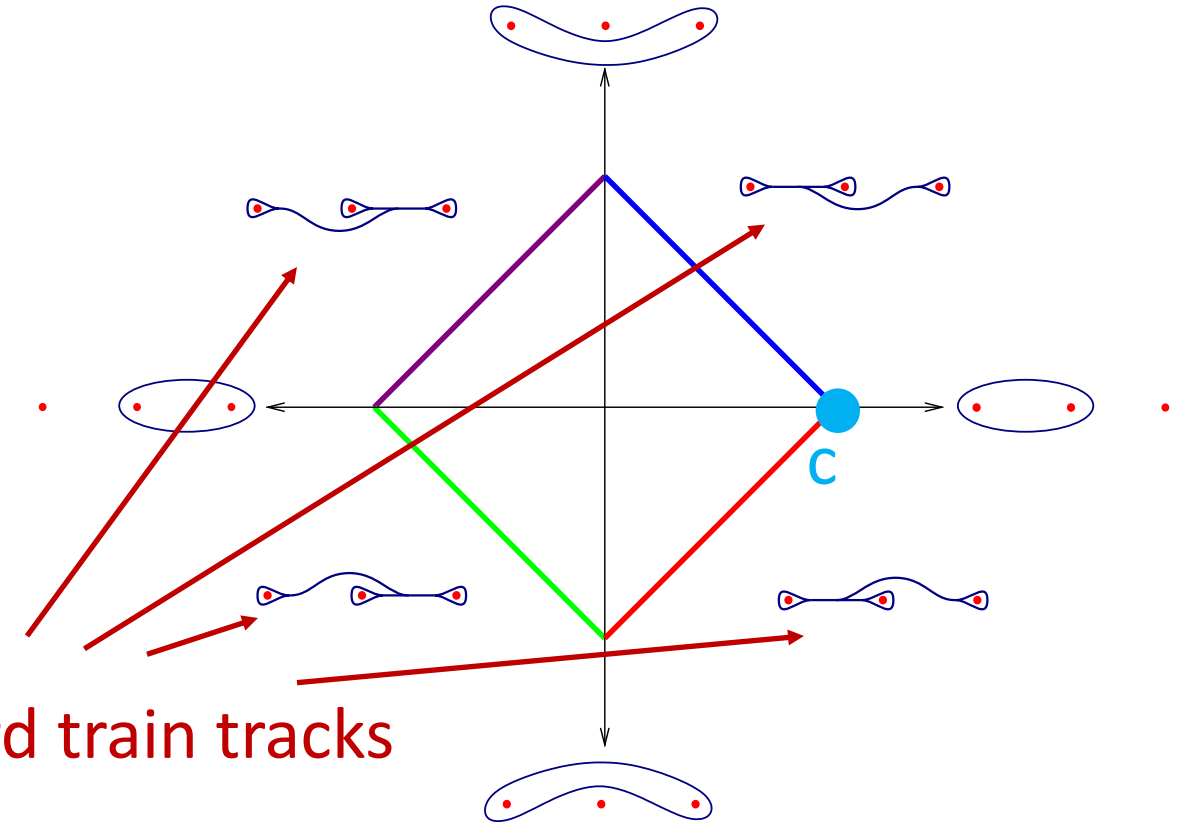
Linear charts on MF = train tracks



measured foliation

Algorithm – setup

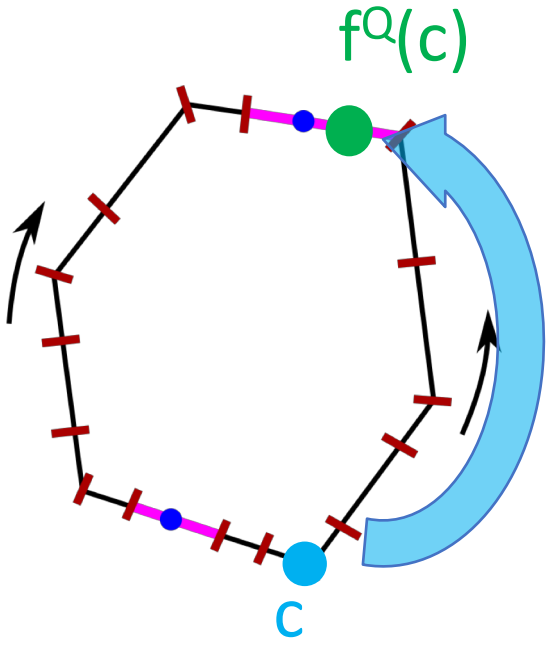
1. Fix a cell decomposition of PMF.
2. Fix some c in PMF.



Standard train tracks

Algorithm v1 (cubic)

1. Check if finite order.
2. Compute $f^Q(c)$ for some Q = O(N).
3. Compute the acting matrix at $f^Q(c)$.
4. Compute eigenvalues and eigenvectors.
5. Check if the eigenvecors correspond to reducing curves, filling or non-filling pA foliations.



Cubic time

Thm (MSY): We get into the attracting linear piece in O(N) iterations.

Cubic time

Thm (MSY): We get into the attracting linear piece in O(N) iterations.

Fact: Each iteration takes O(N^2) time.

Cubic time

Thm (MSY): We get into the attracting linear piece in O(N) iterations.

Fact: Each iteration takes O(N^2) time.

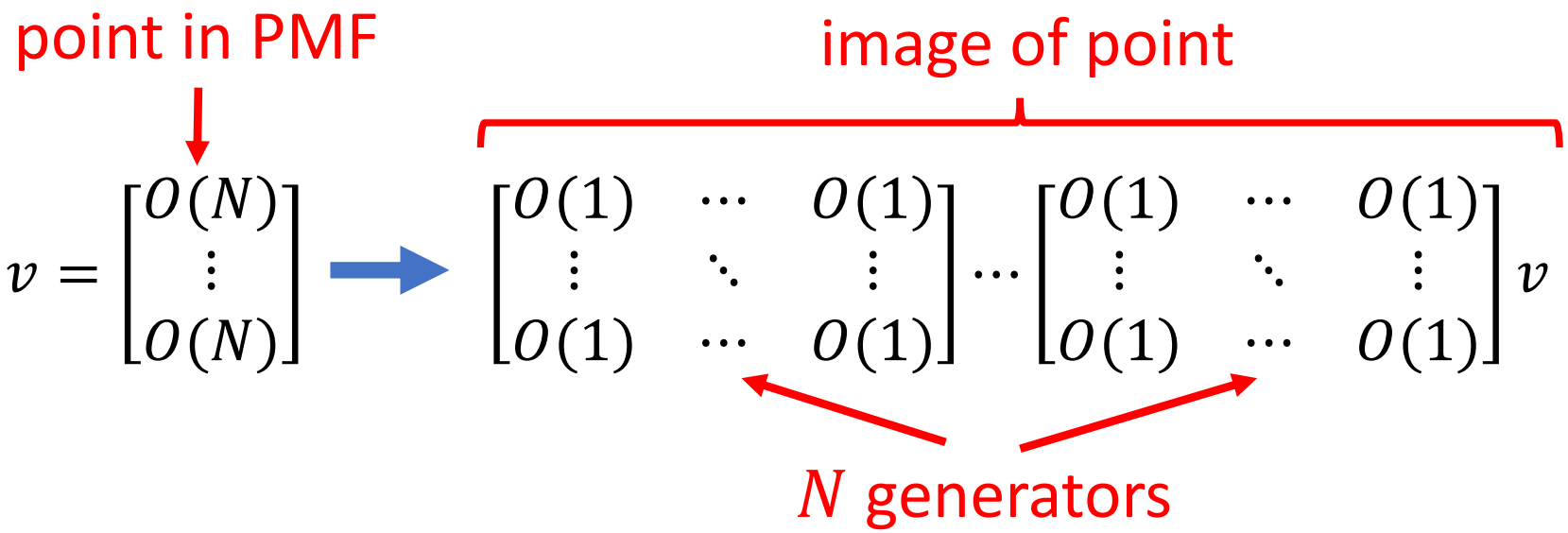
point in PMF

$$v = \begin{bmatrix} O(N) \\ \vdots \\ O(N) \end{bmatrix}$$

Cubic time

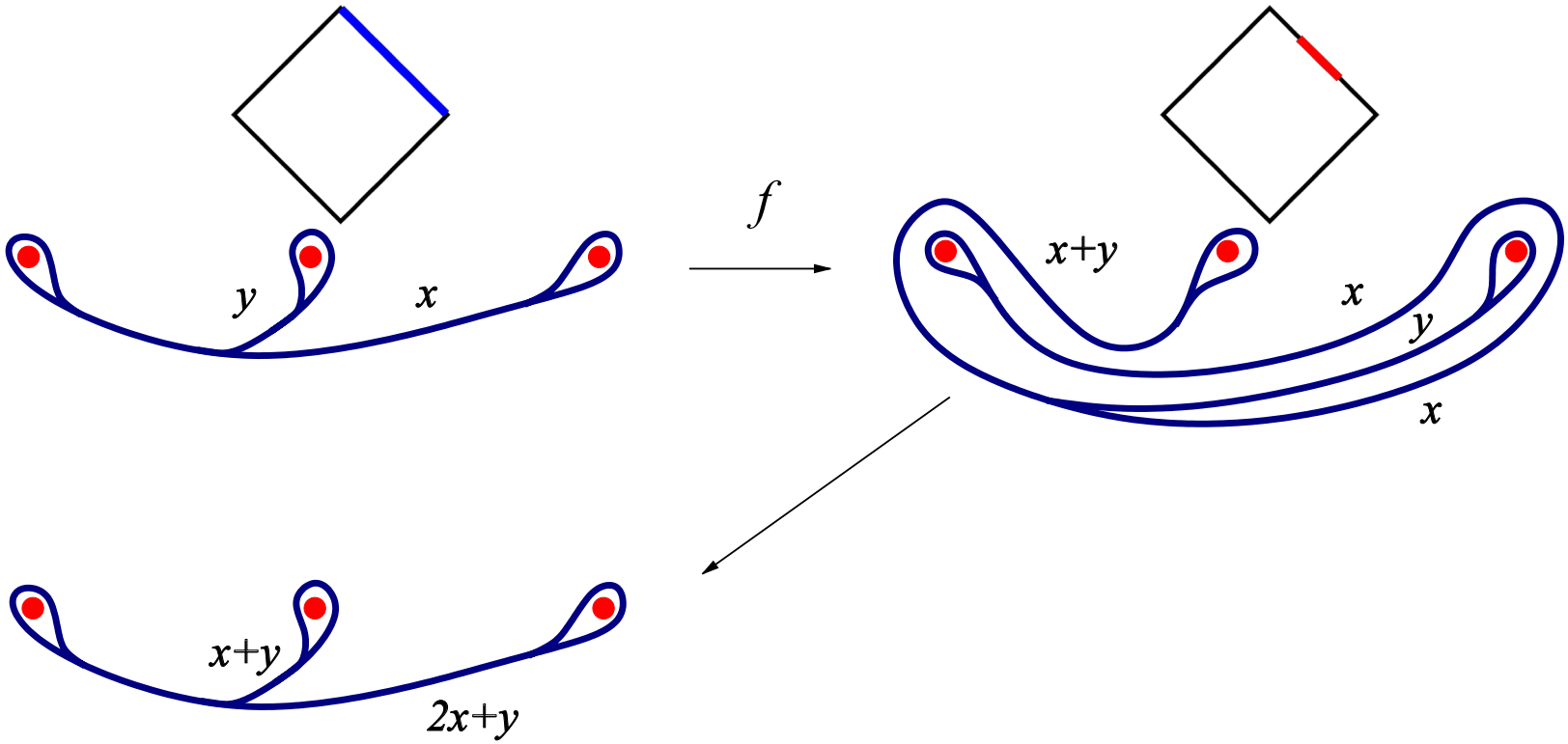
Thm (MSY): We get into the attracting linear piece in O(N) iterations.

Fact: Each iteration takes O(N^2) time.

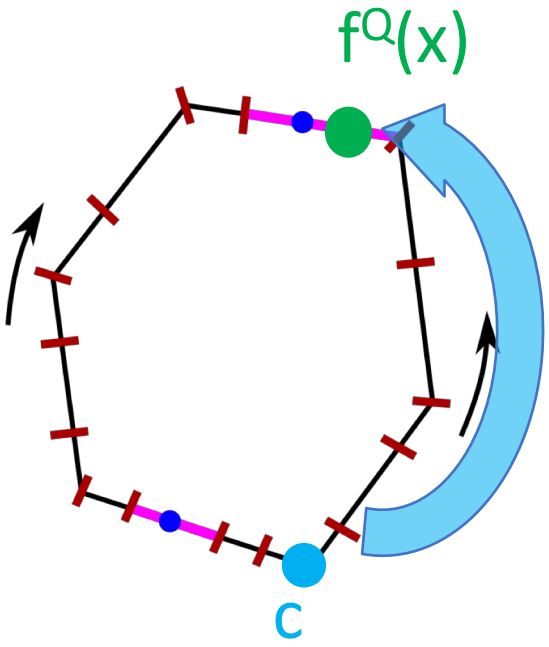


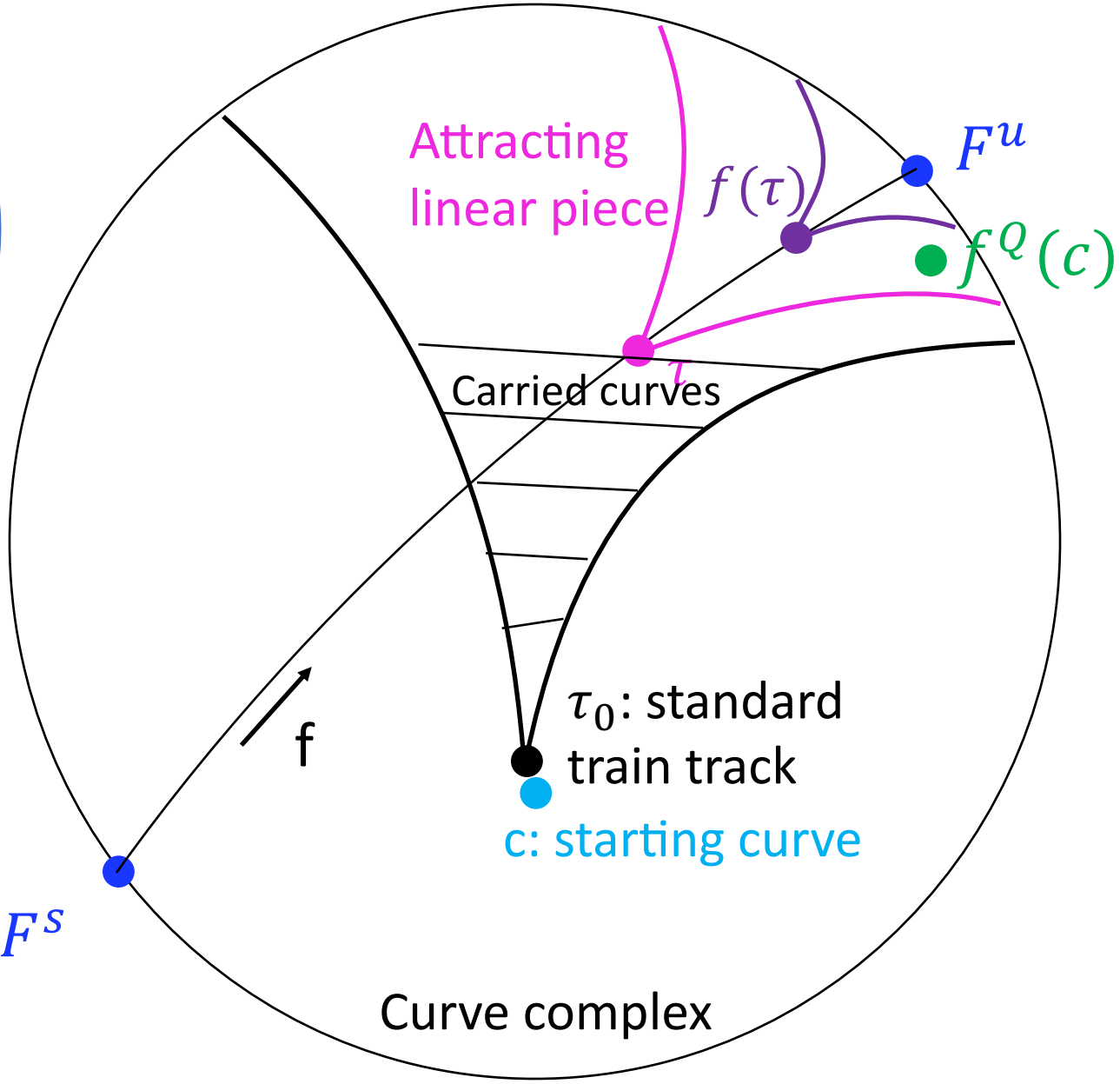
Why do we get into the attracting linear piece in O(N) iterations?

Linearly compatibity = carrying



Attracting linear piece: Invariant train track carried by a standard train track.

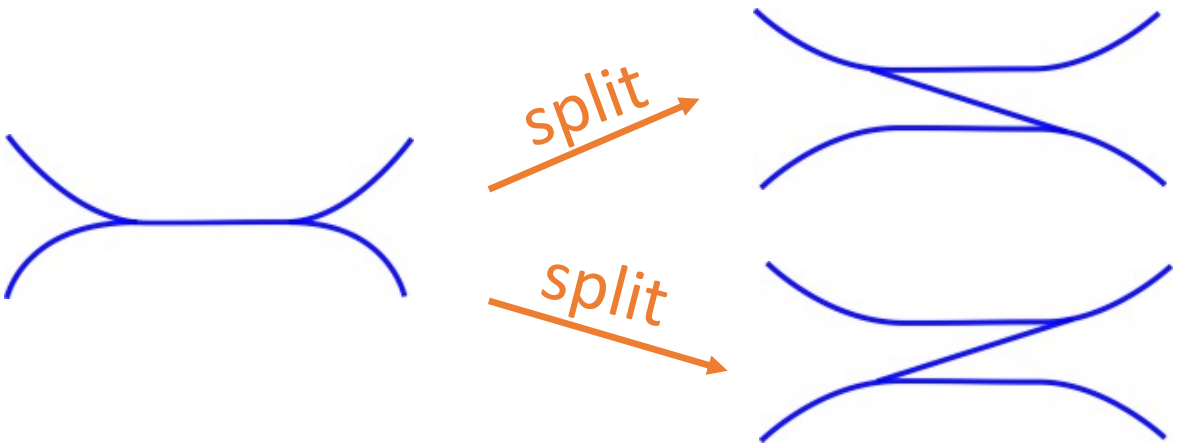




Q: How deep is τ inside τ_0?

Splittings

A way to create train tracks carried.

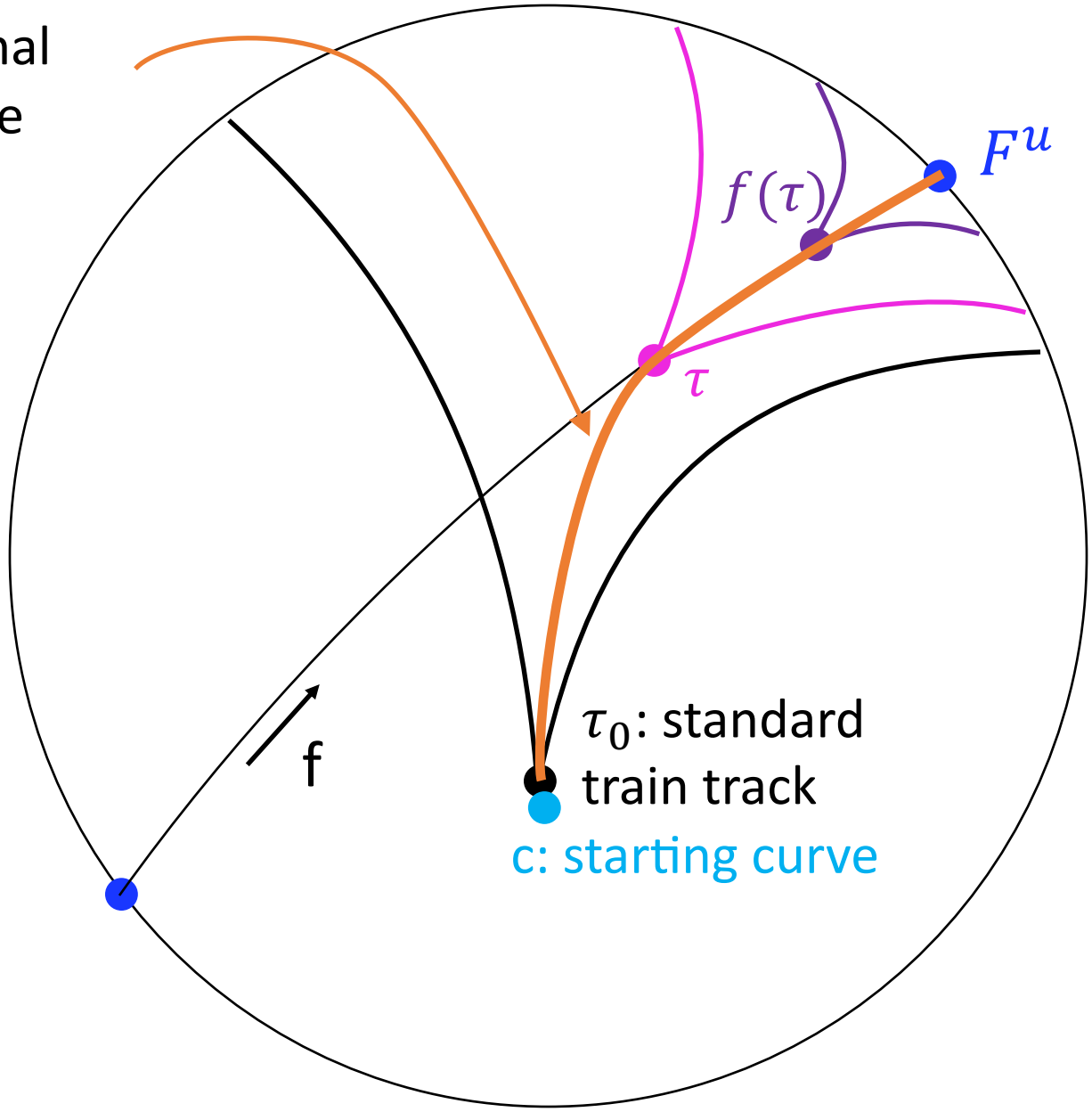


Agol (‘11): maximal splitting sequence

Bell (‘14): preperiodic length is at most O(N) times the periodic length

Masur-Minsky’s nested train track argument:

$c \in f^{-O(N)}(\tau)$

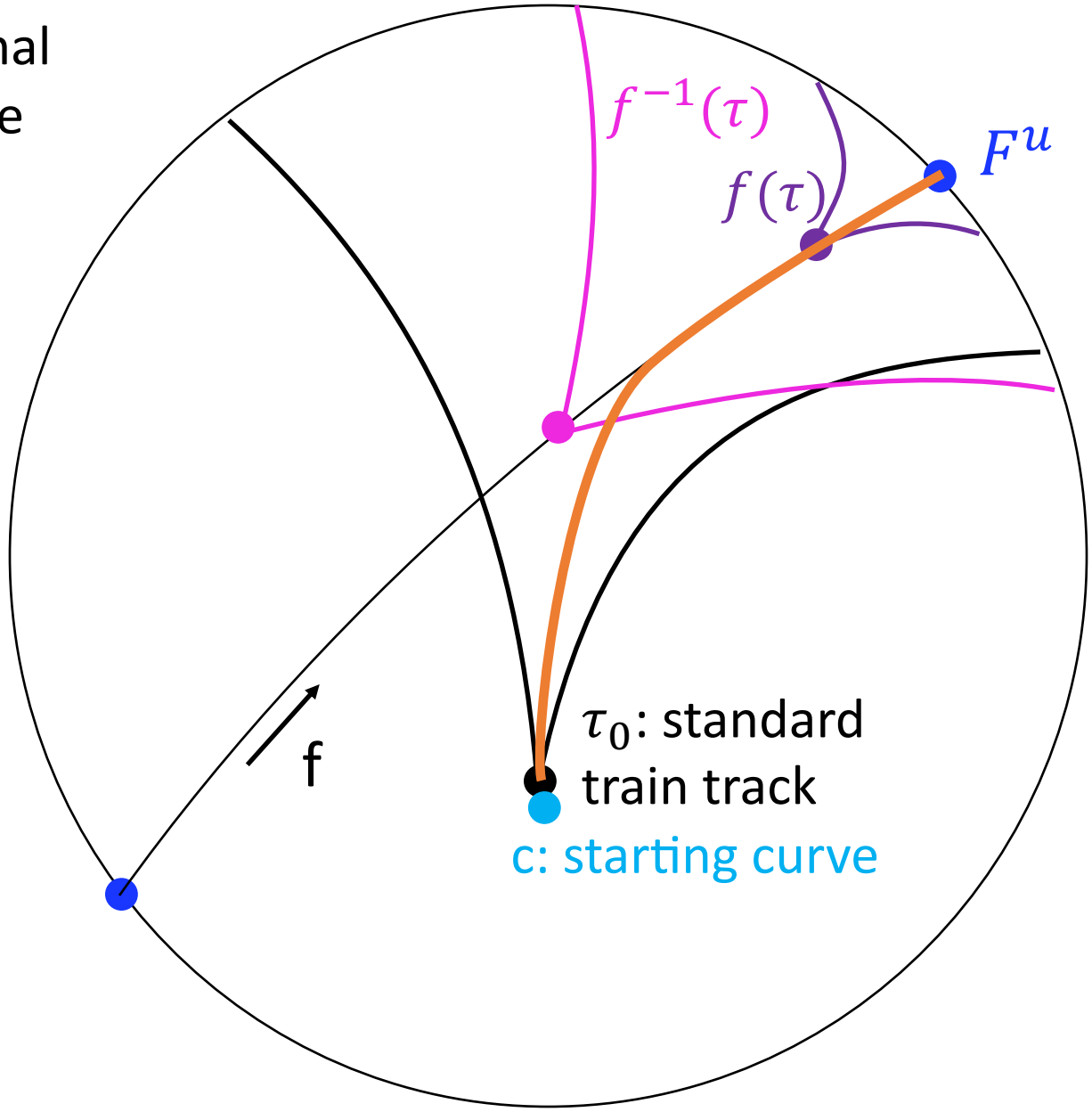


Agol (‘11): maximal splitting sequence

Bell (‘14): preperiodic length is at most O(N) times the periodic length

Masur-Minsky’s nested train track argument:

$c \in f^{-O(N)}(\tau)$

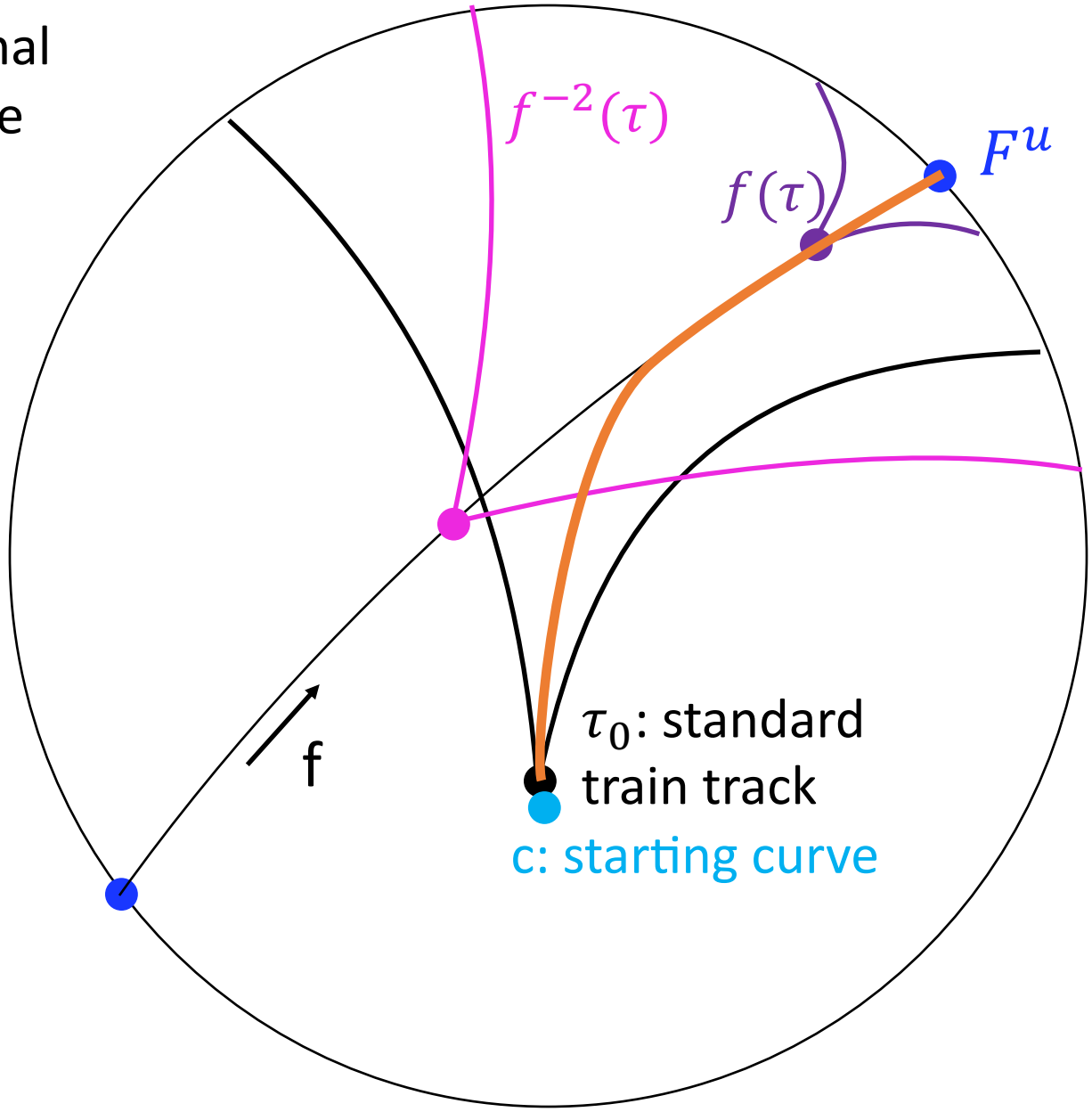


Agol (‘11): maximal splitting sequence

Bell (‘14): preperiodic length is at most O(N) times the periodic length

Masur-Minsky’s nested train track argument:

$c \in f^{-O(N)}(\tau)$

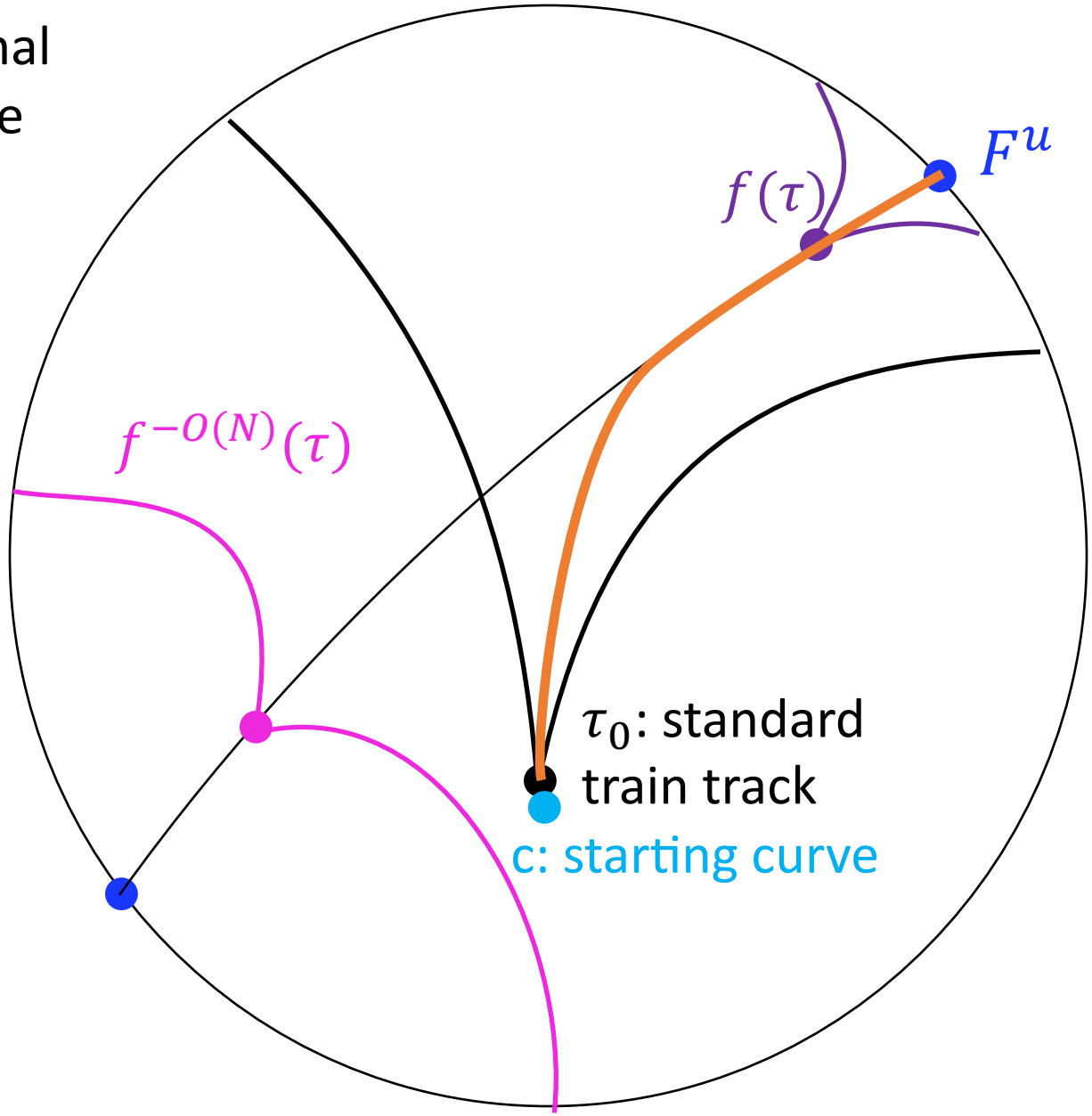


Agol (‘11): maximal splitting sequence

Bell (‘14): preperiodic length is at most O(N) times the periodic length

Masur-Minsky’s nested train track argument:

$c \in f^{-O(N)}(\tau)$



Agol (‘11): maximal splitting sequence

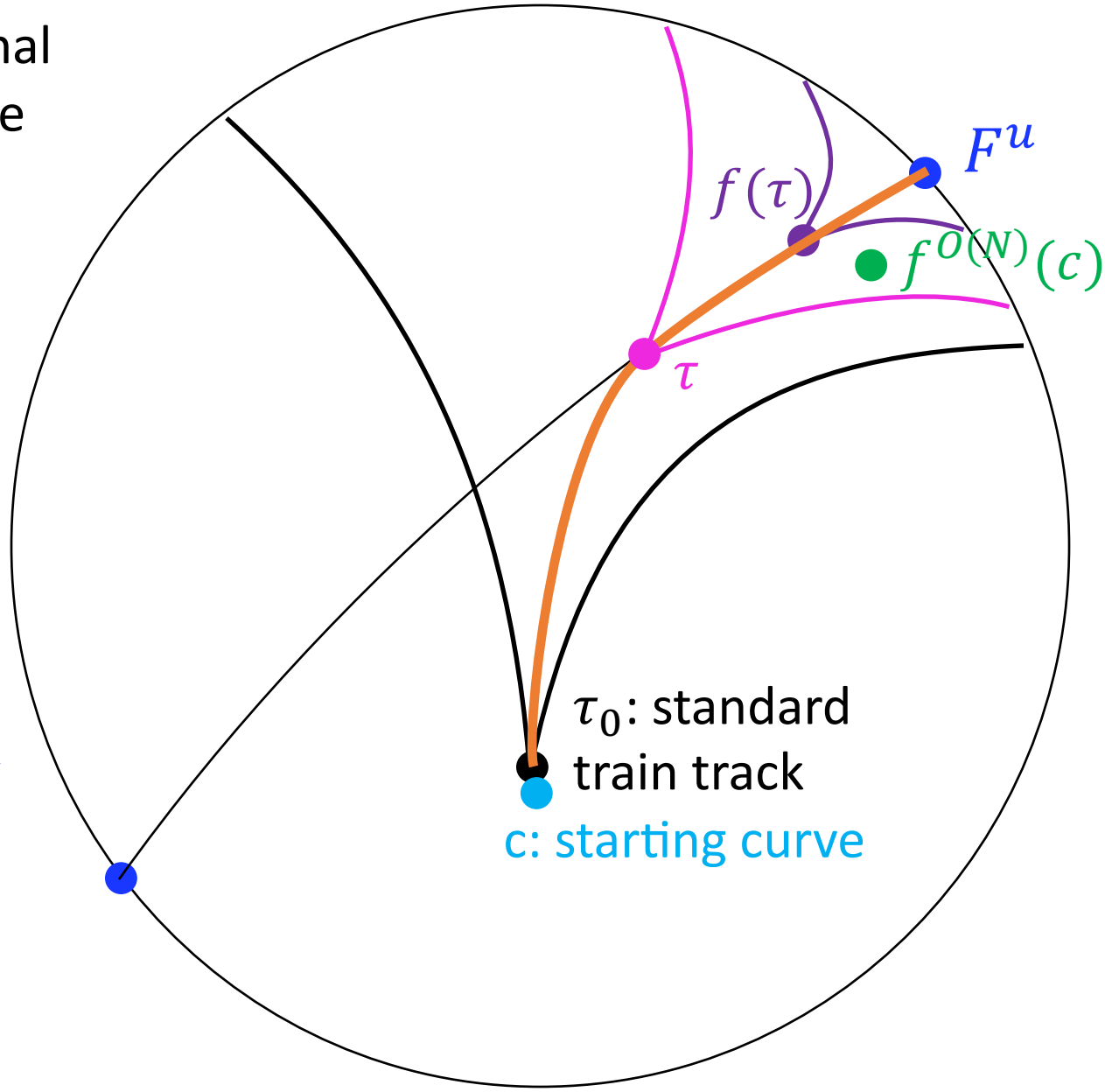
Bell (‘14): preperiodic length is at most O(N) times the periodic length

Masur-Minsky’s nested train track argument:

$c \in f^{-O(N)}(\tau)$

so

$f^{O(N)}(c) \in \tau$



Main tool in reducible case

Masur-Mosher-Schleimer (‘12): splitting sequences and progress made in subsurfaces.

Subtlety 1

Computing the eigenvalues quickly is problematic because of Bell-Schleimer. Iterative methods are all exponential.

Solution: Use Newton's method to find the largest eigenvalue quickly.

Subtlety 2

For a reducible with multiple pseudo-Anosov components, how do we find all pA stretch factors, not just the largest one?

Subtlety 2

For a reducible with multiple pseudo-Anosov components, how do we find all pA stretch factors, not just the largest one?

Solution: Factor the characteristic polynomial to irreducible factors

Subtlety 2

For a reducible with multiple pseudo-Anosov components, how do we find all pA stretch factors, not just the largest one?

Solution: Factor the characteristic polynomial to irreducible factors (cubic time by Lenstra-Lenstra-Lovász's seminal 1982 work on lattice reductions).

Subtlety 2

For a reducible with multiple pseudo-Anosov components, how do we find all pA stretch factors, not just the largest one?

Solution: Factor the characteristic polynomial to irreducible factors (cubic time by Lenstra-Lenstra-Lovász's seminal 1982 work on lattice reductions). Then run Newton's method on each factor to find the largest root.

The bound for the number of iterations is only O(N), but we think O(1) should be possible as well.

Algorithm v2

Old tools

- Agol (11)
- Beloi (14)
- Masur-Mosher-Schleimer (12),

New tools

- Flat surfaces
- Slope
- Veering triangulations (Agol, Guéritaud, Minsky-Taylor)

Slope

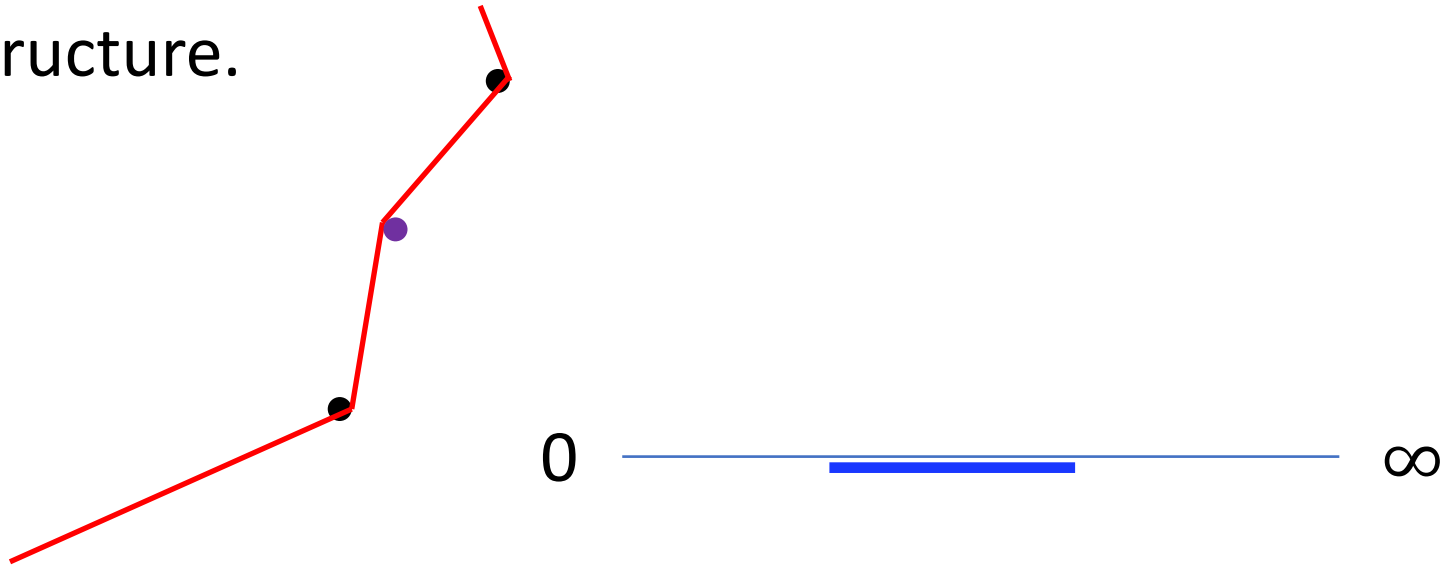
$F^u, F^s \rightarrow$ flat structure.

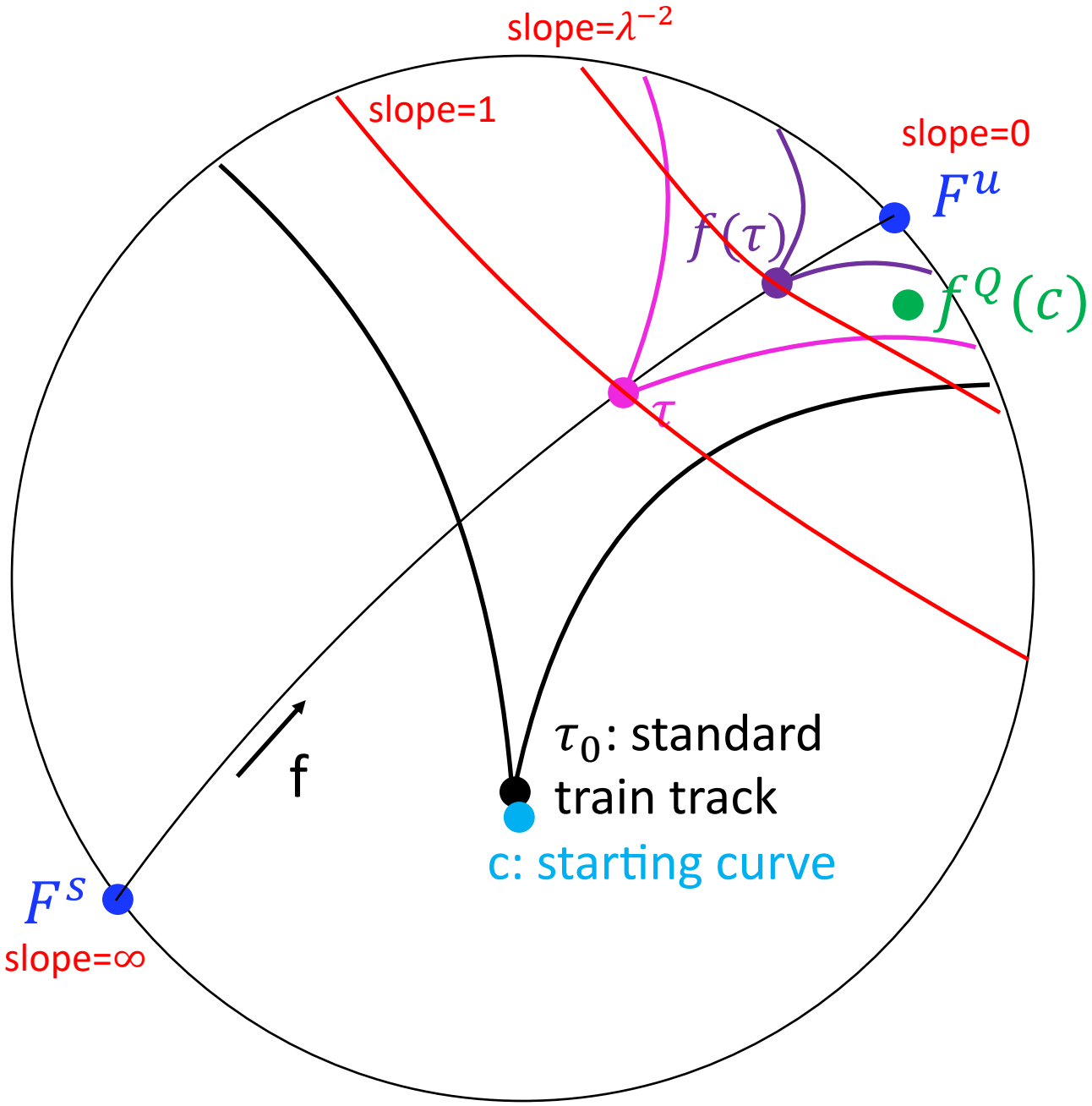
F^u - horizontal foliation, F^s - vertical foliation

Slope

$F^u, F^s \rightarrow$ flat structure.
F^u - horizontal foliation, F^s - vertical foliation

The slope of a curve is the (range of) absolute values of slopes of saddle connections when pulled tight in flat structure.




slope=λ^{-2}
slope=1
slope=0
F^u
$f(\tau)$
$f^Q(c)$
τ
f
τ_0: standard
train track
c: starting curve
F^s
slope=∞

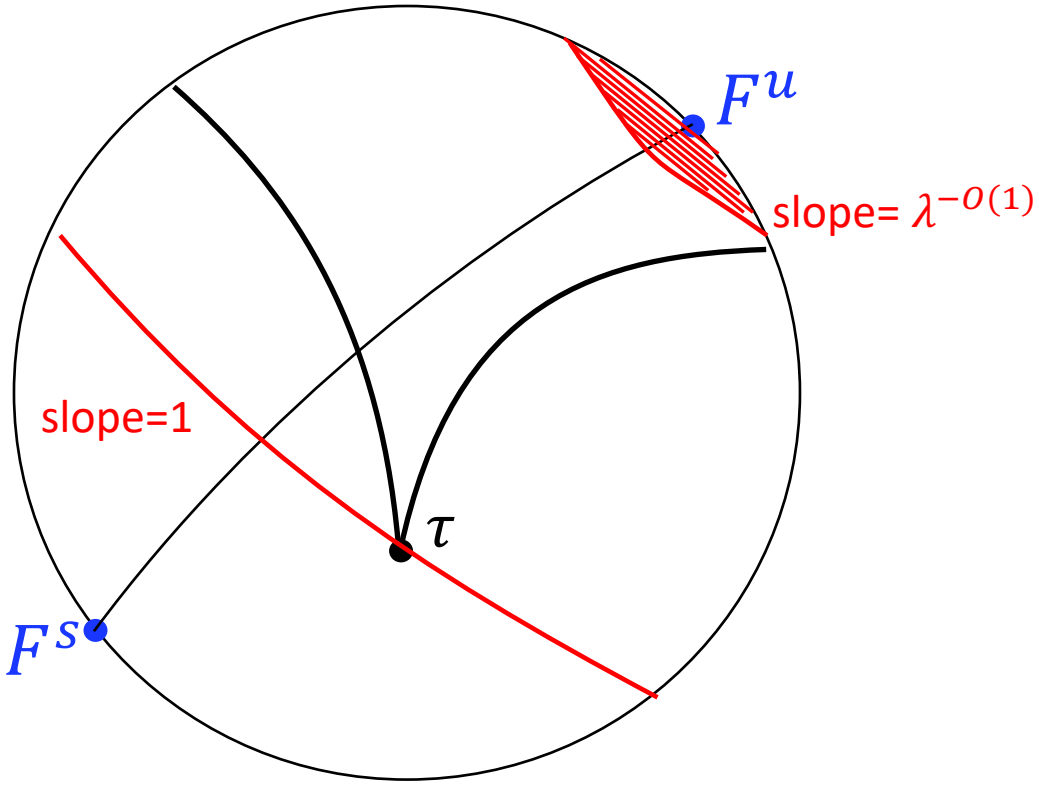
Fact 1 (obvious): $slope\big(f(c)\big) = \lambda^{-2} slope(c)$

Fact 1 (obvious): $slope\big(f(c)\big) = \lambda^{-2} slope(c)$

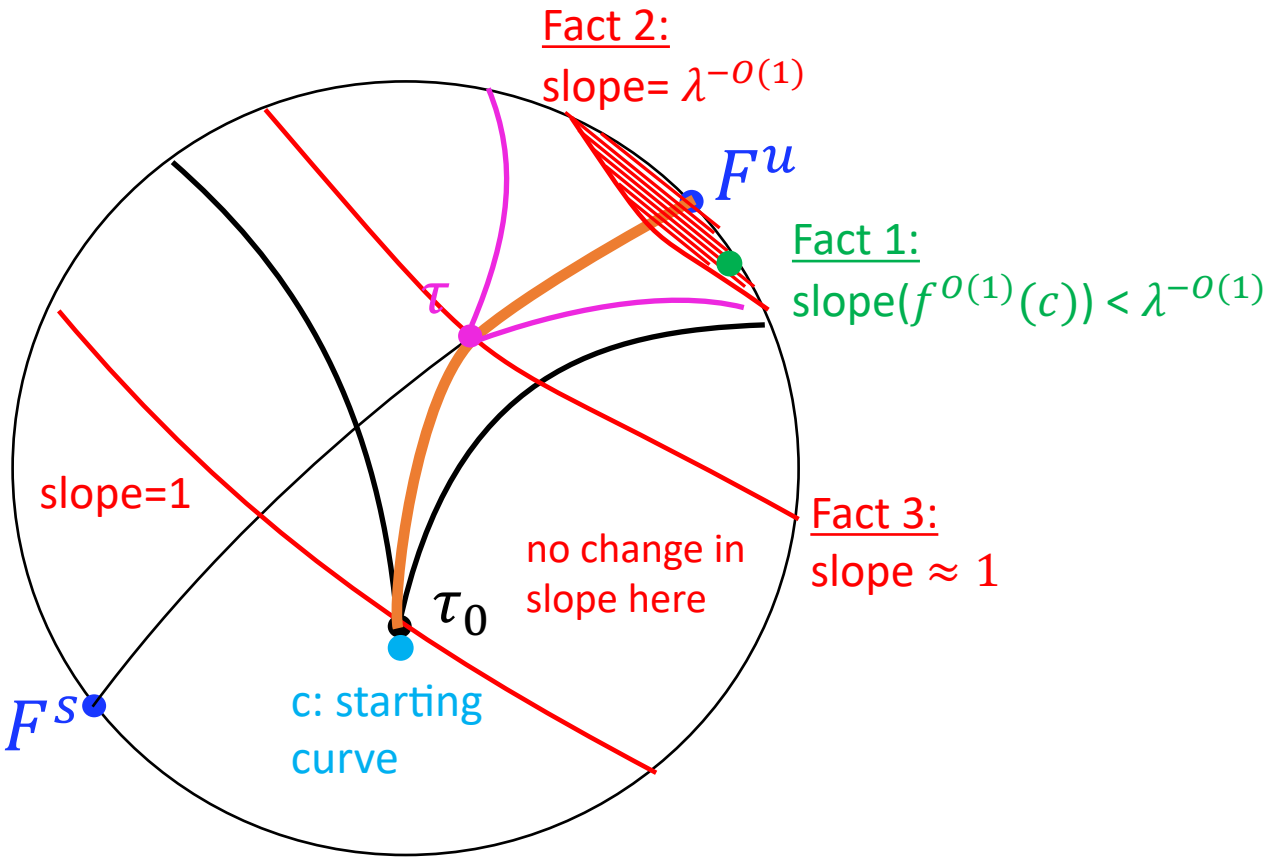
Fact 2 (MSY): If τ carries F^{u} and $slope(c) \ll slope(\tau)$, then the curve c is carried on (a diagonal extension of) τ.

Fact 1 (obvious): $slope\big(f(c)\big) = \lambda^{-2} slope(c)$

Fact 2 (MSY): If τ carries F^{u} and $slope(c) \ll slope(\tau)$, then the curve c is carried on (a diagonal extension of) τ.



Fact 3 (MSY): For any splitting sequence starting at τ_0 and carrying F^u, the slope can only decrease once the train tracks are invariant under some $f^{O(1)}$.



Issue

Lenstra-Lenstra-Lovász's cubic time polynomial factorization does not cut it anymore.

Solution: van Hoeij-Novocin's quadratic time improvement ('12).

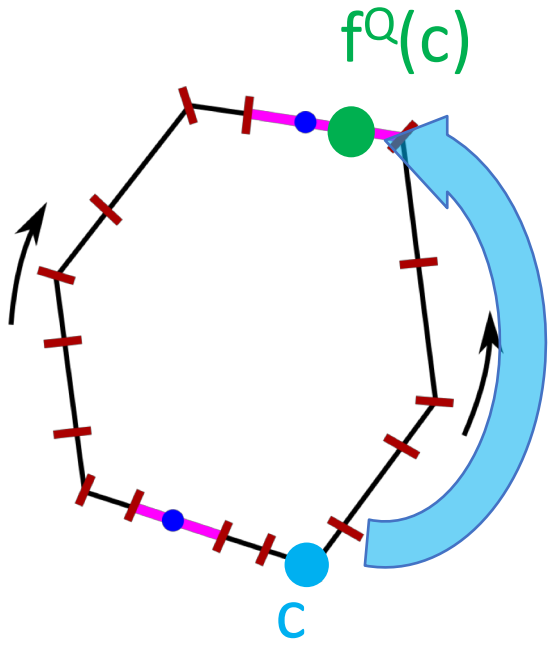
Issue

Trouble with more than 1-dimensional eigenspaces. E.g., it is possible that 1 is an eigenvalue with high multiplicity, but there is only one reducing curve.

Solution: Use train track splittings to identify the portion of the eigenspace that stays positive under splittings.

Algorithm v2 (quadratic)

1. Check if finite order.
2. Compute $f^Q(c)$ for some $Q = O(1)$.
3. Compute the acting matrix at $f^Q(c)$.
4. Compute eigenvalues and eigenvectors.
5. Check if the eigenvecors correspond to reducing curves, filling or non-filling pA foliations.



There must be a better way

There must be a better way

- We shouldn’t have to rely on deep results from computational number theory.

There must be a better way

- We shouldn't have to rely on deep results from computational number theory.

- Dealing with the eigenspaces this way is quite technical and annoying.

Realization:

There must be a better way

- We shouldn't have to rely on deep results from computational number theory.

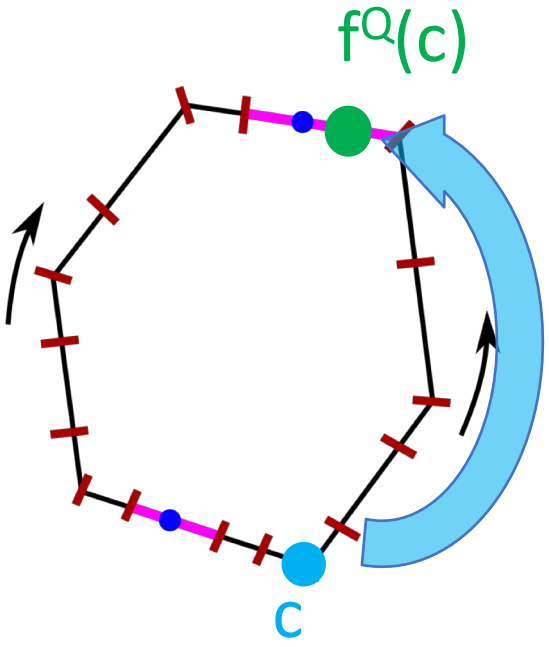
- Dealing with the eigenspaces this way is quite technical and annoying.

Realization: It is a mistake to resort to linear algebra. By doing that, we lose geometric information that we need to recover later by other methods.

PMF
Train&racks
Linear algebra

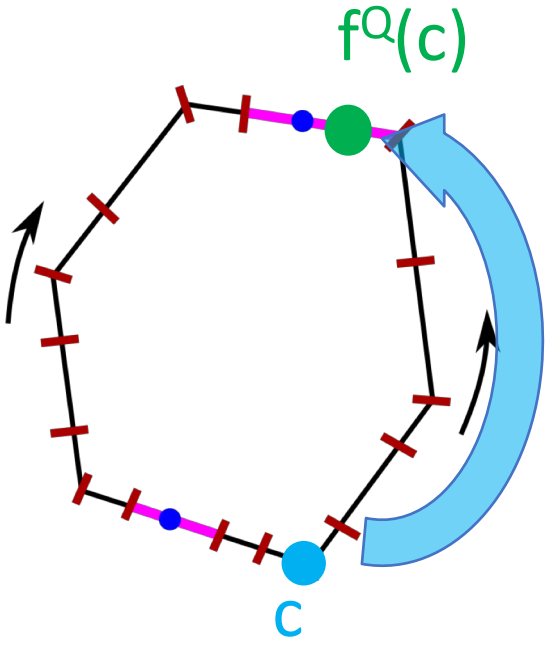
Algorithm v3

Goals



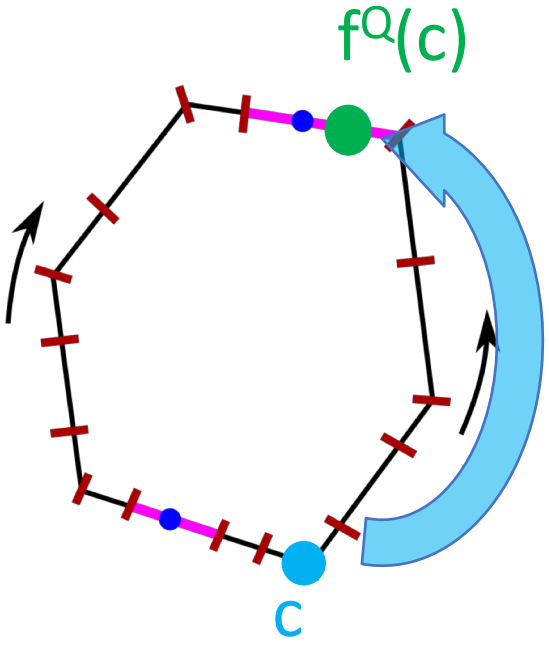
Goals

1. Find the train track of the attracting linear region.

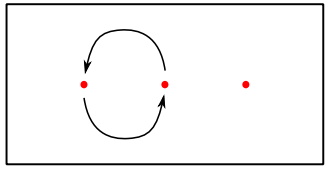


Goals

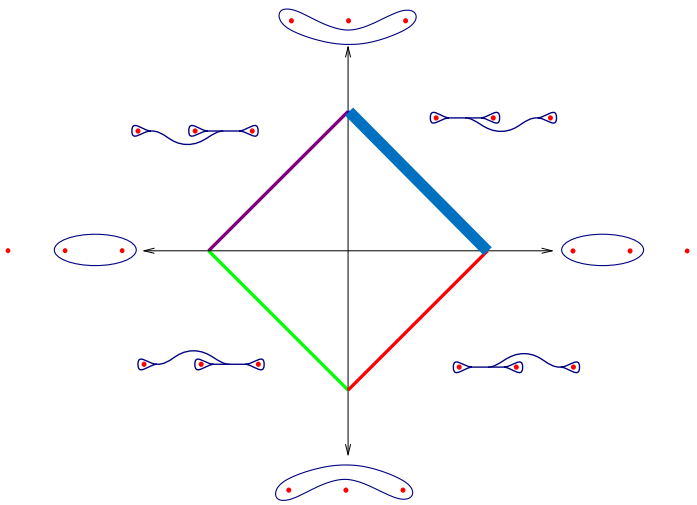
1. Find the train track of the attracting linear region.
2. Compute the train track self-map instead of just the matrix.

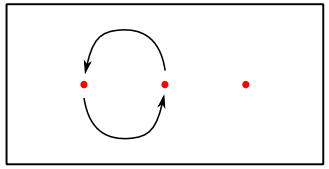


Natural splitting sequences

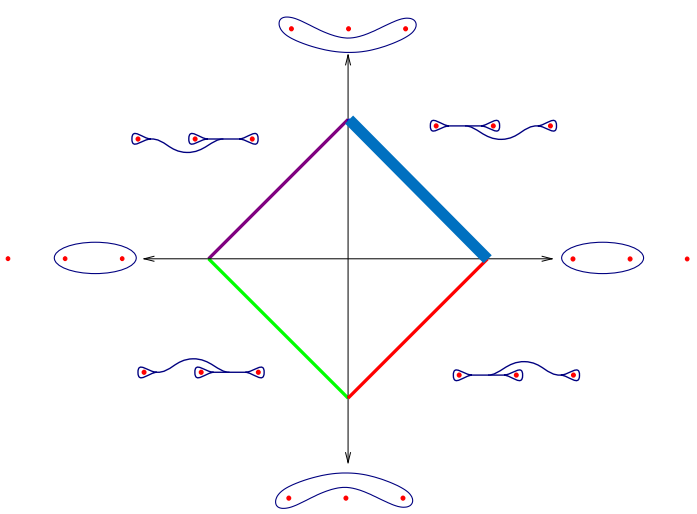


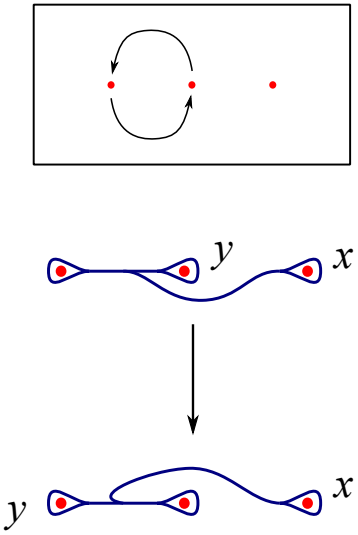
Natural splitting sequences



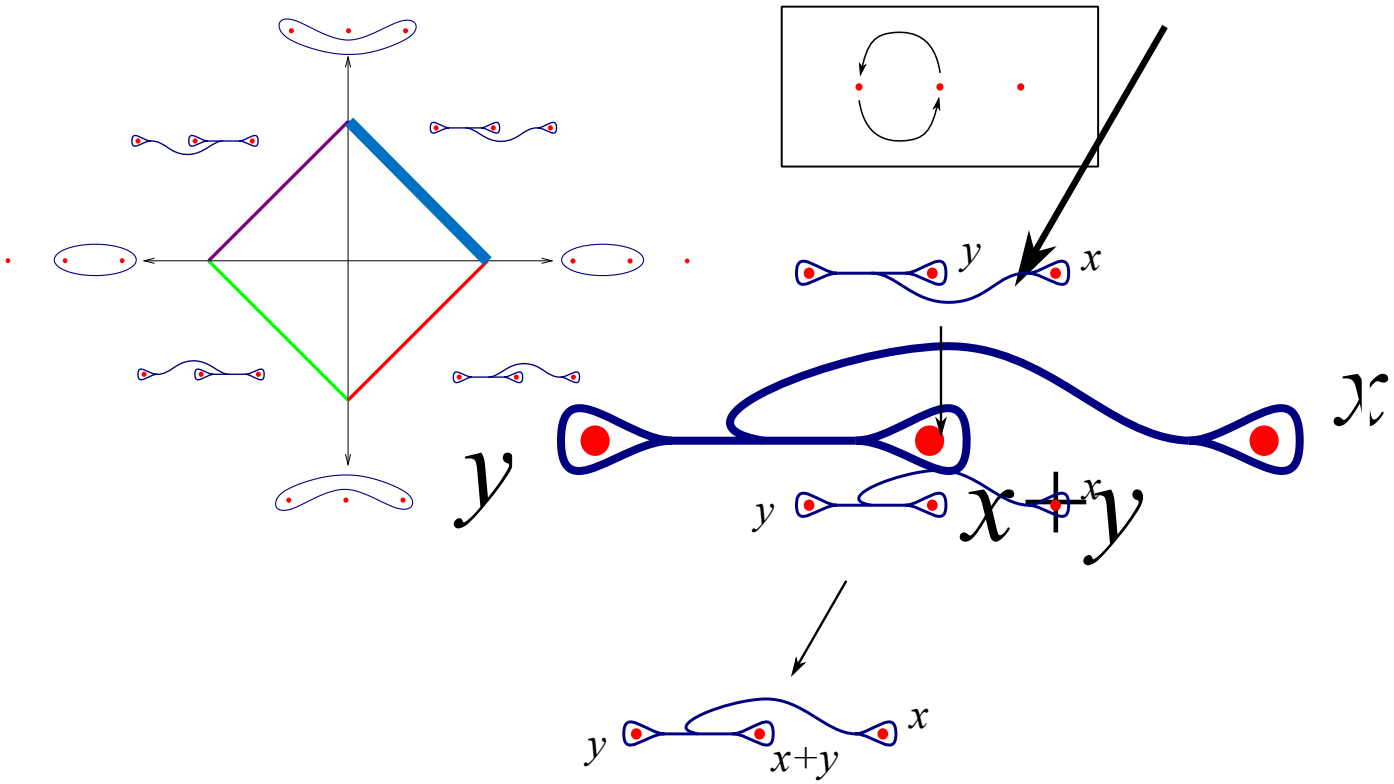


Natural splitting sequences

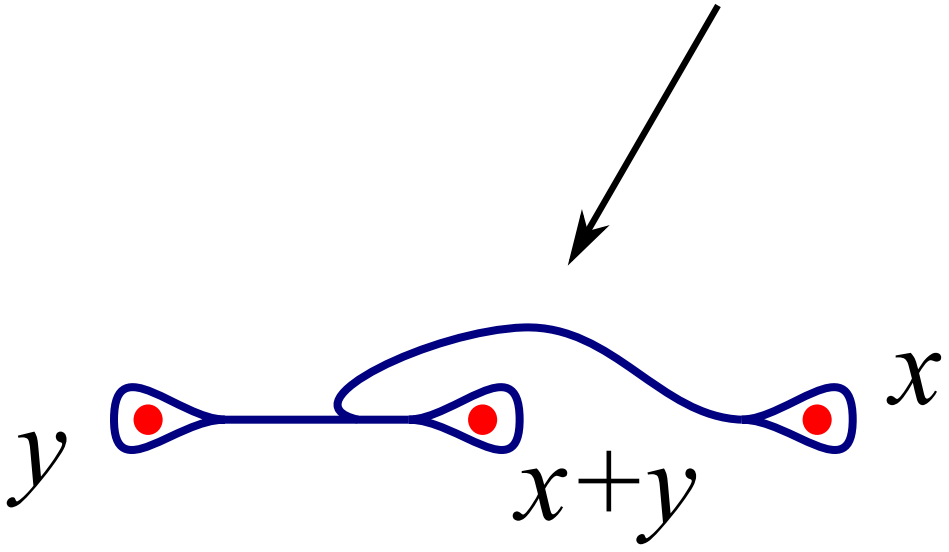




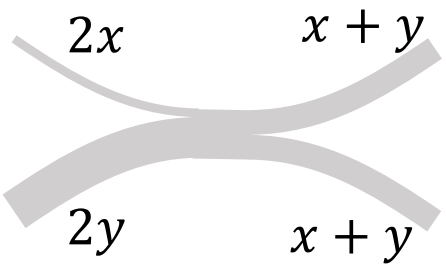
Natural splitting sequences



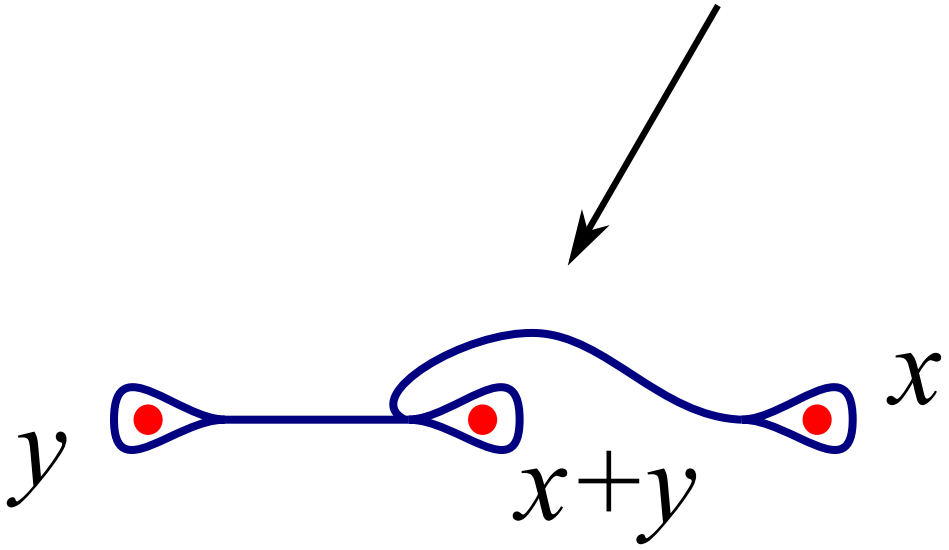
Natural splitting sequences



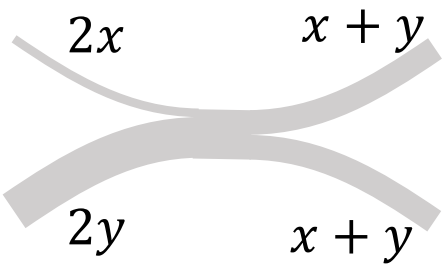
$y \geq x$



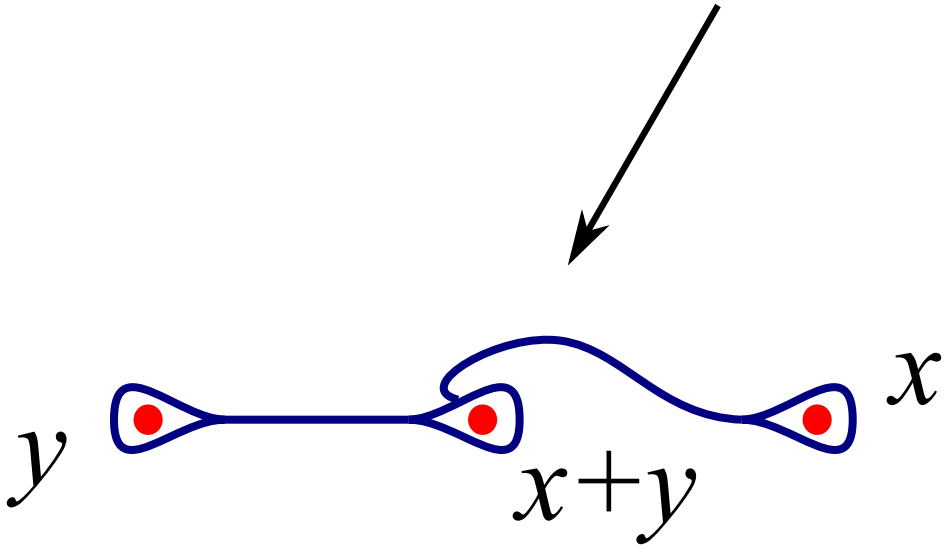
Natural splitting sequences



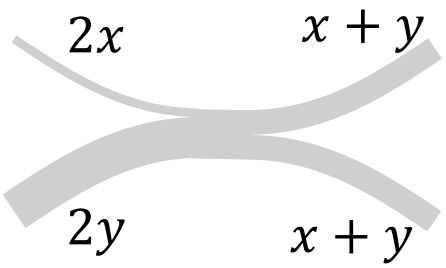
$y \geq x$



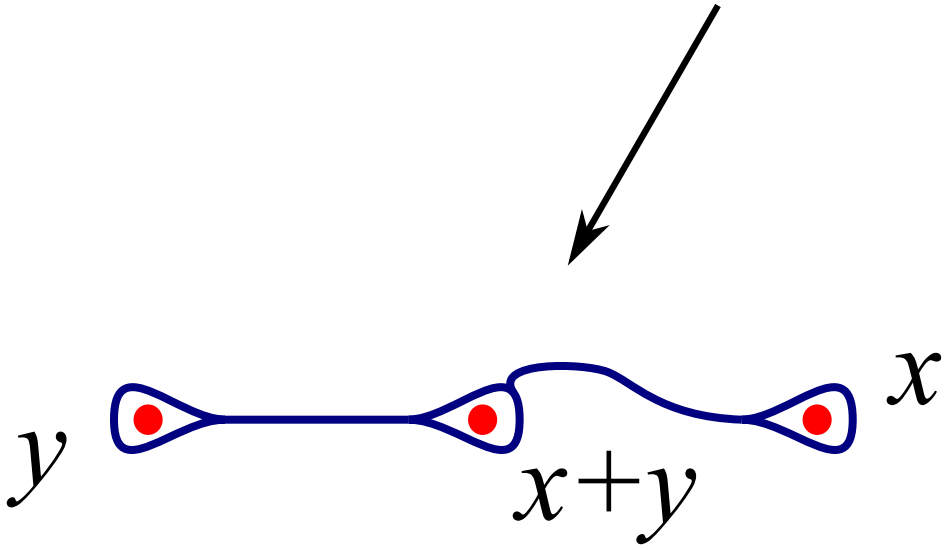
Natural splitting sequences



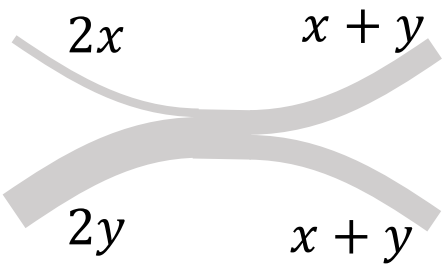
$y \geq x$



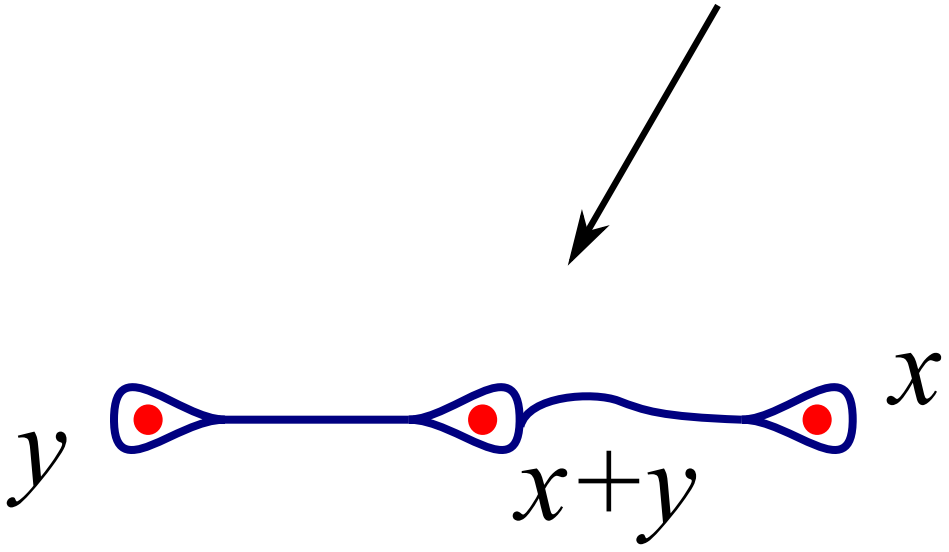
Natural splitting sequences



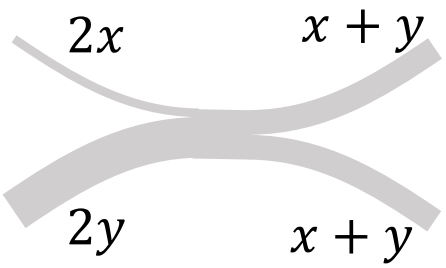
$y \geq x$



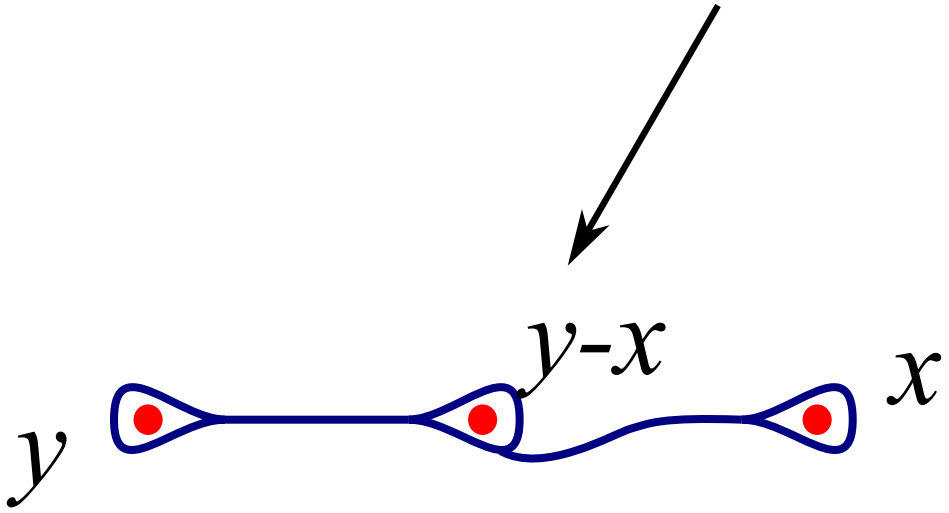
Natural splitting sequences



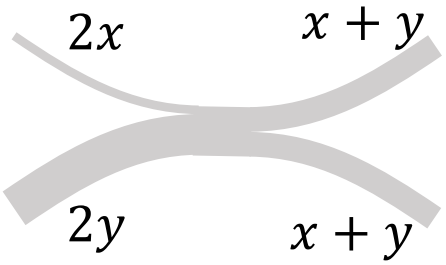
$y \geq x$



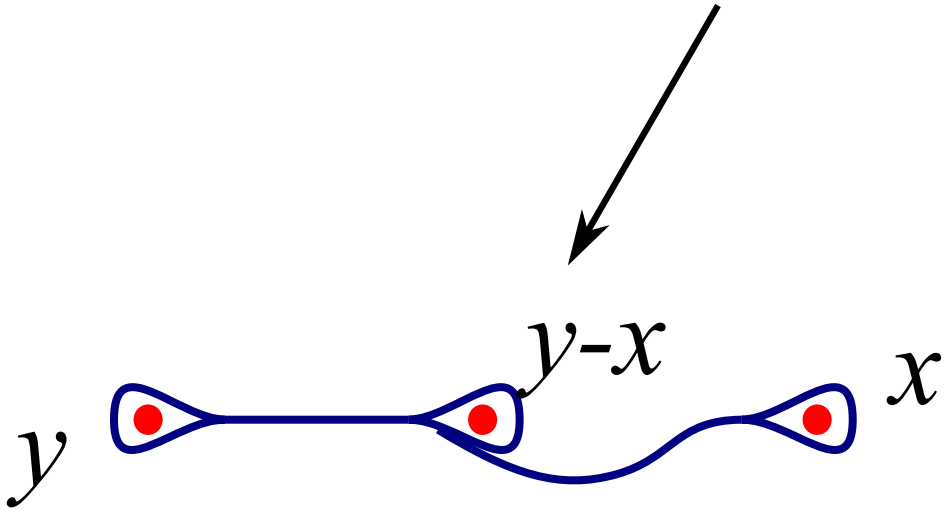
Natural splitting sequences



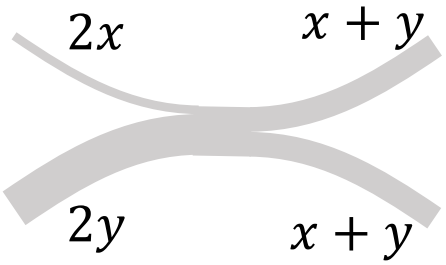
$y \geq x$



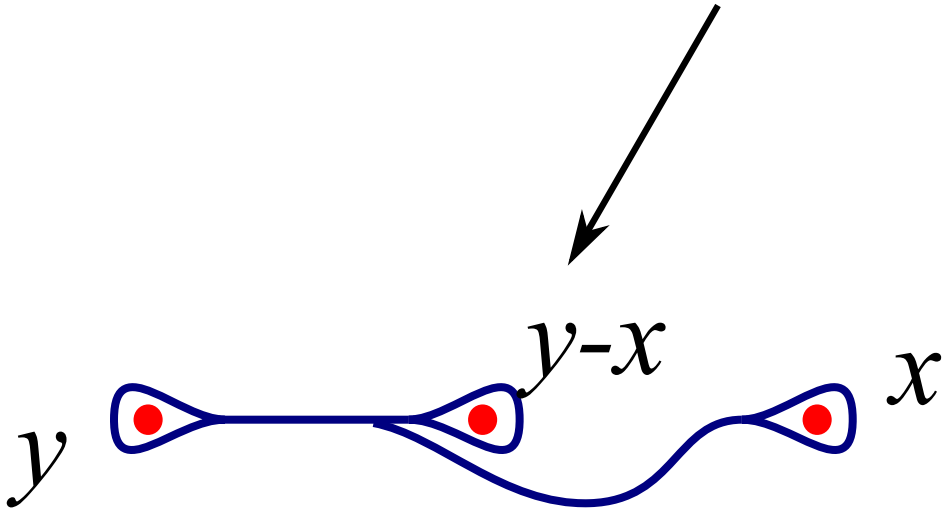
Natural splitting sequences



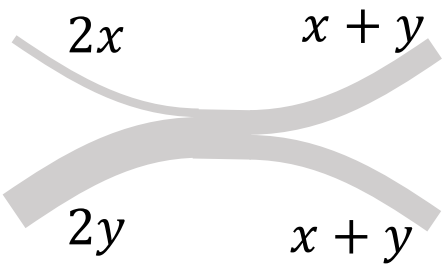
$y \geq x$



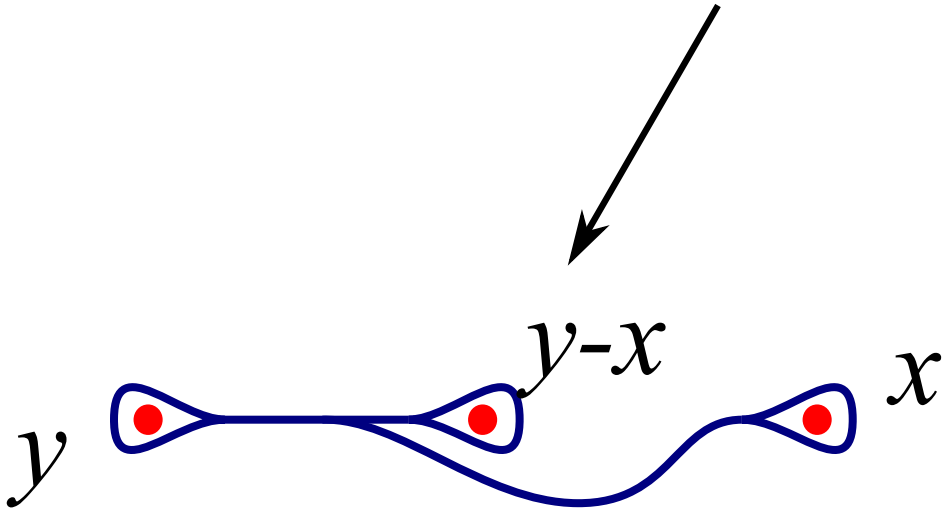
Natural splitting sequences



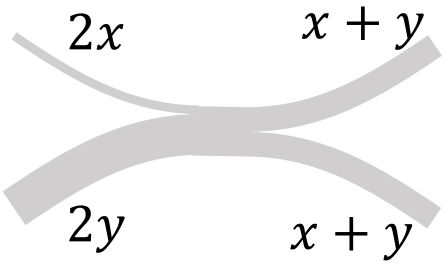
$y \geq x$



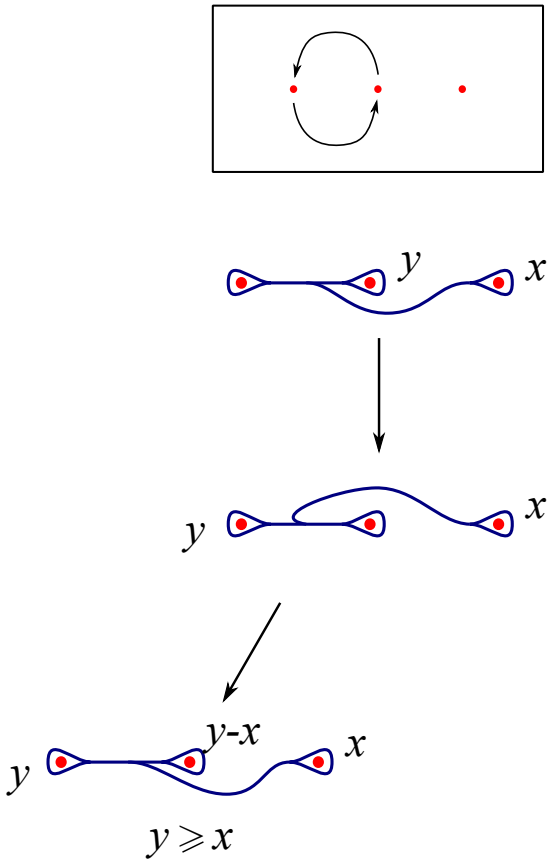
Natural splitting sequences



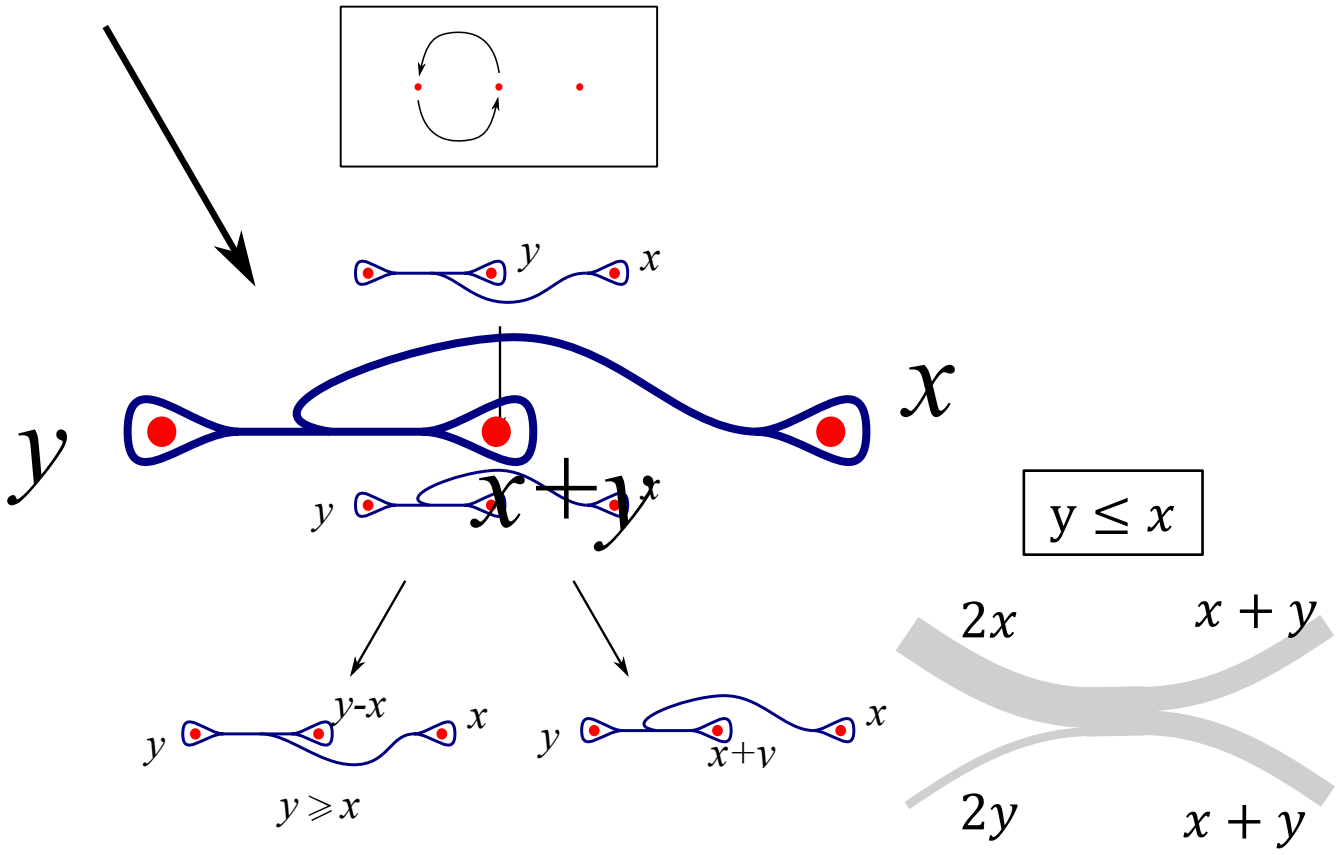
$y \geq x$



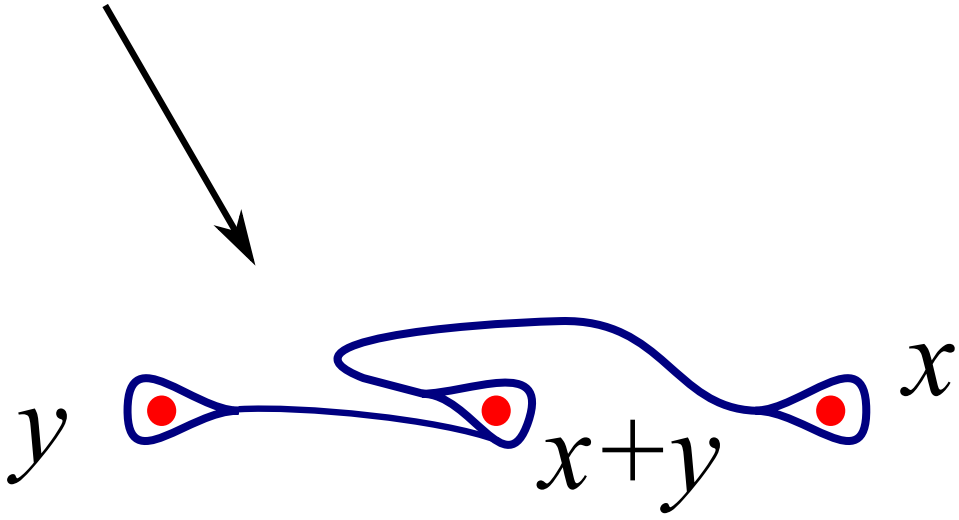
Natural splitting sequences



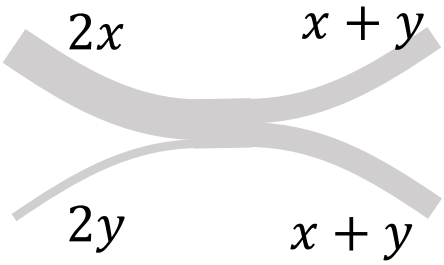
Natural splitting sequences



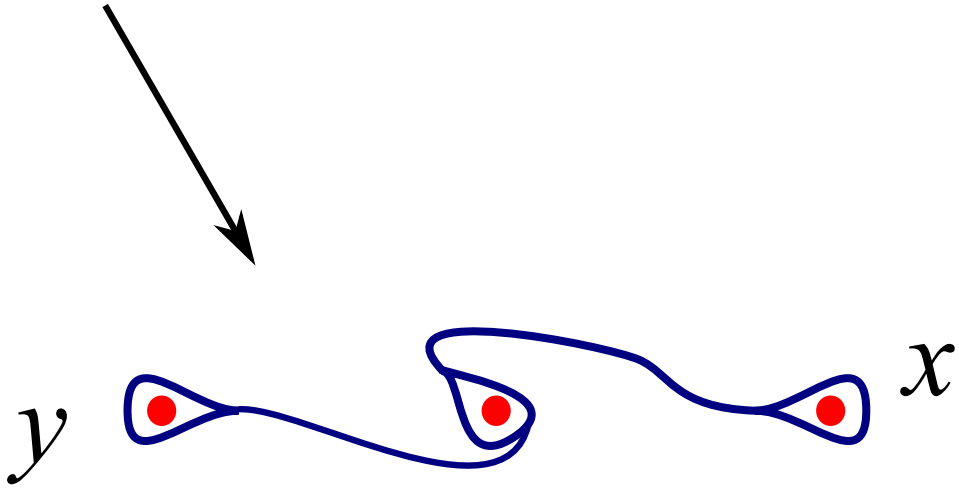
Natural splitting sequences



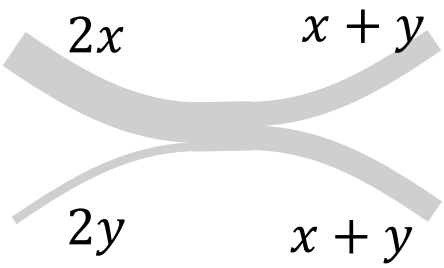
$y \leq x$



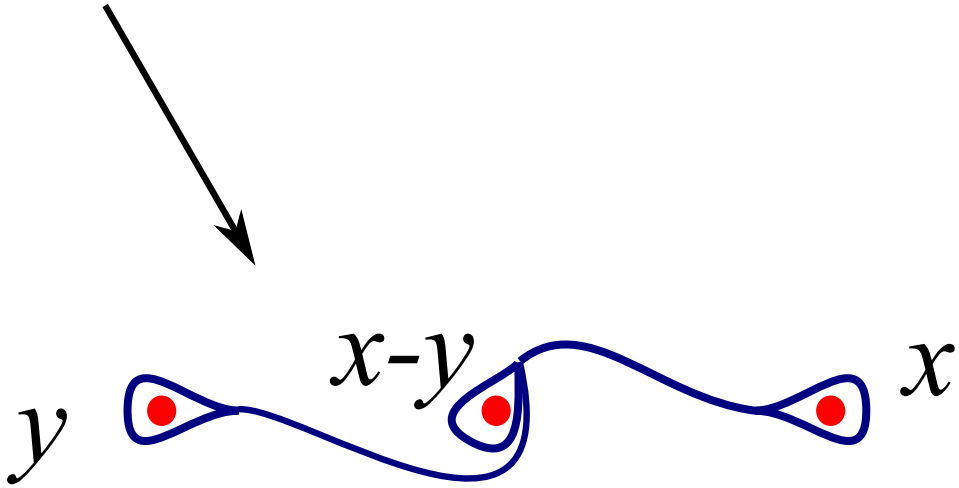
Natural splitting sequences



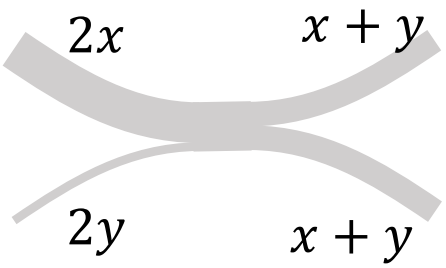
$y \leq x$



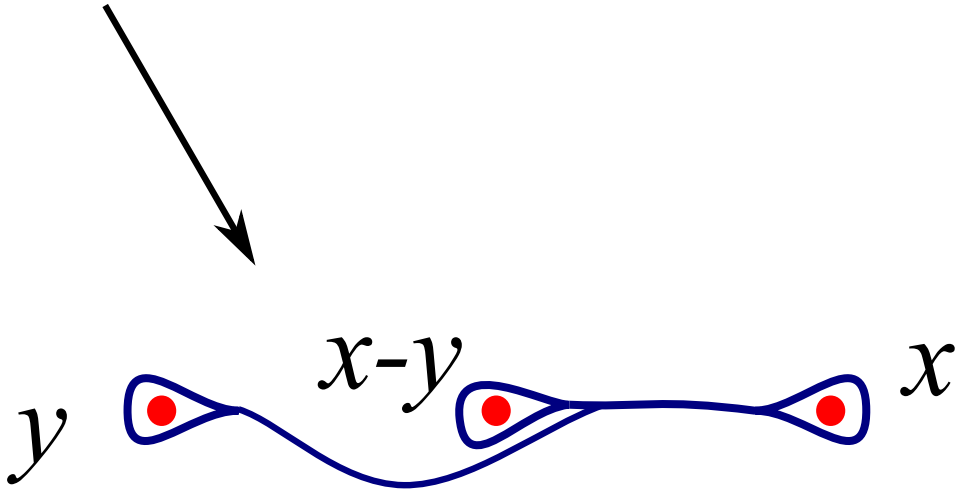
Natural splitting sequences



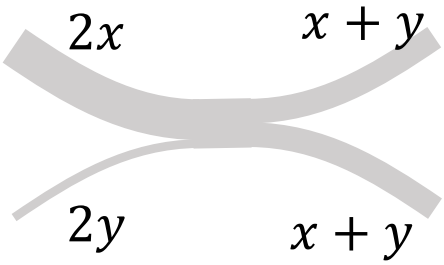
$y \leq x$



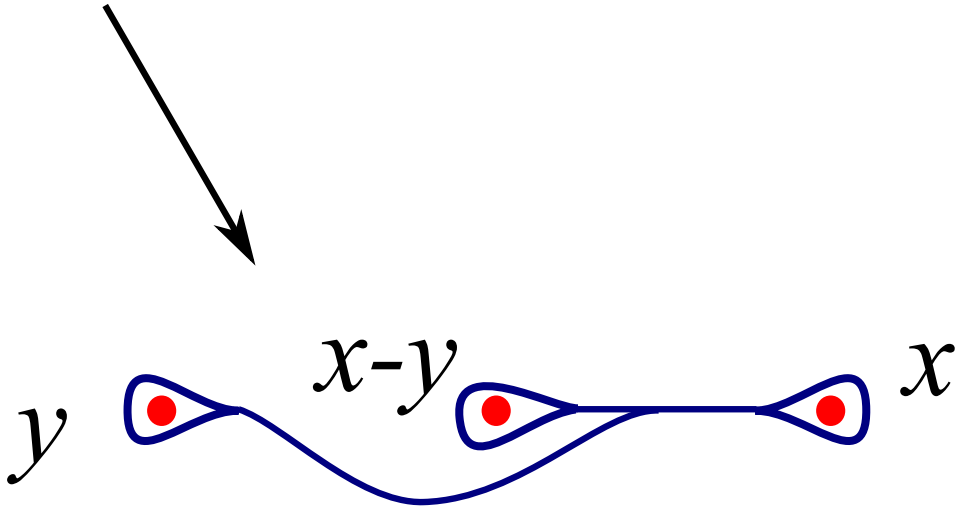
Natural splitting sequences



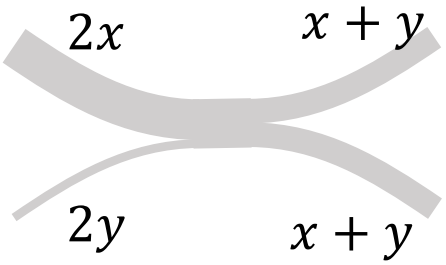
$y \leq x$



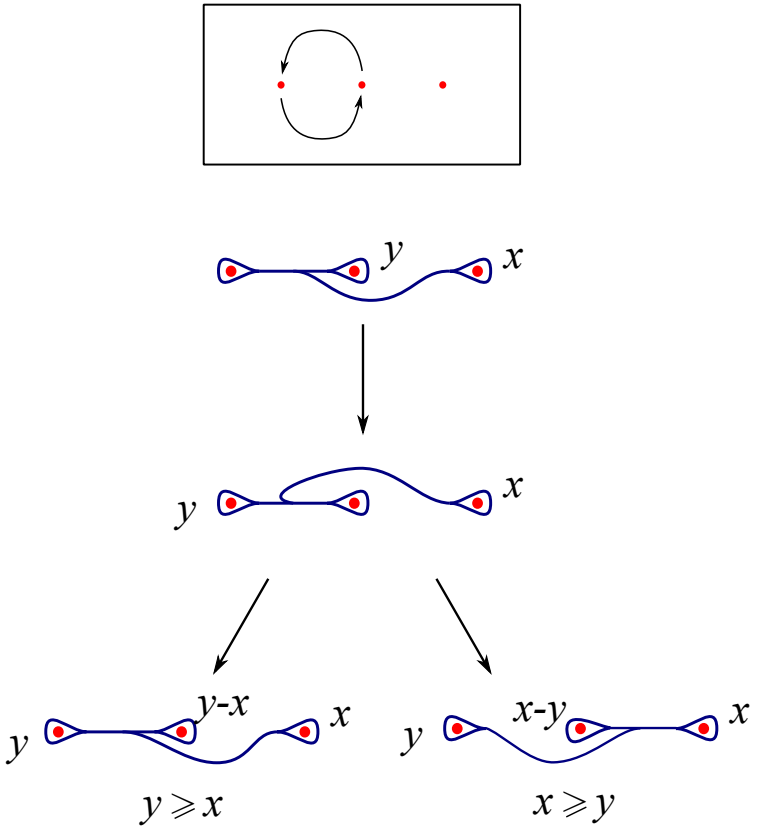
Natural splitting sequences



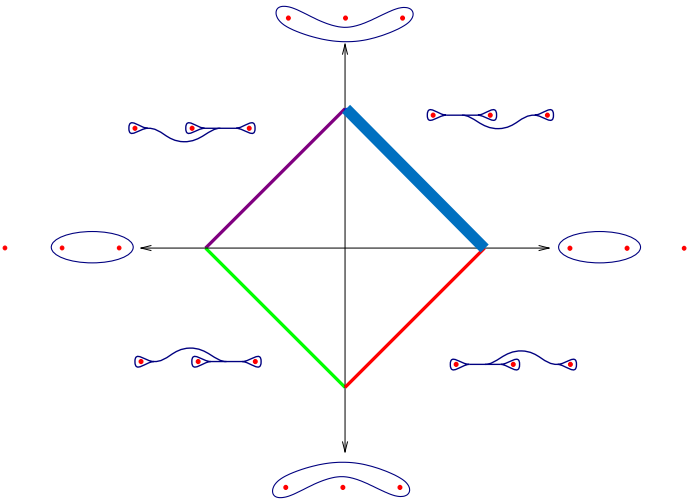
$y \leq x$

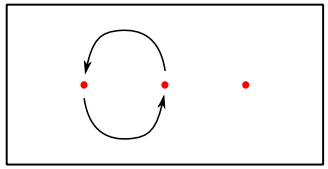


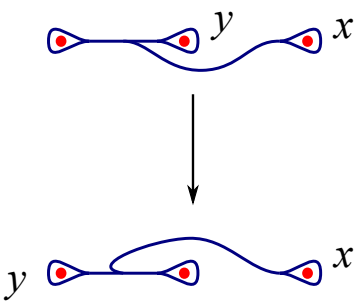
Natural splitting sequences

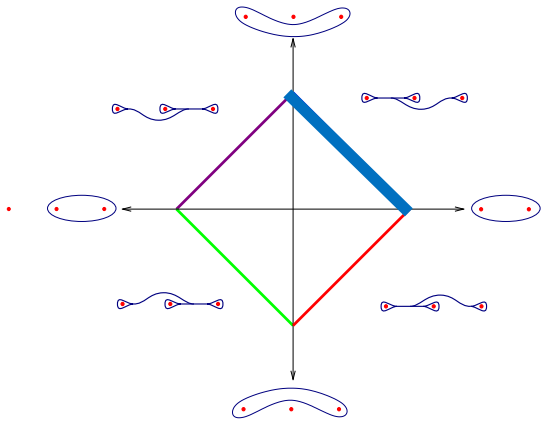


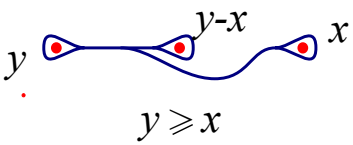
Natural splitting sequences

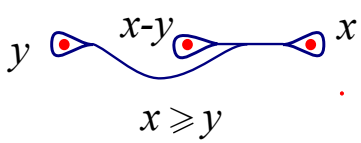


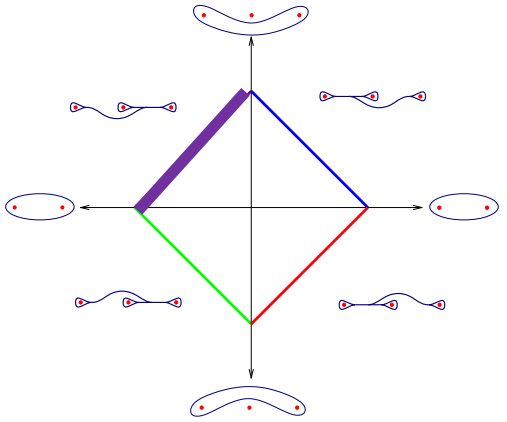




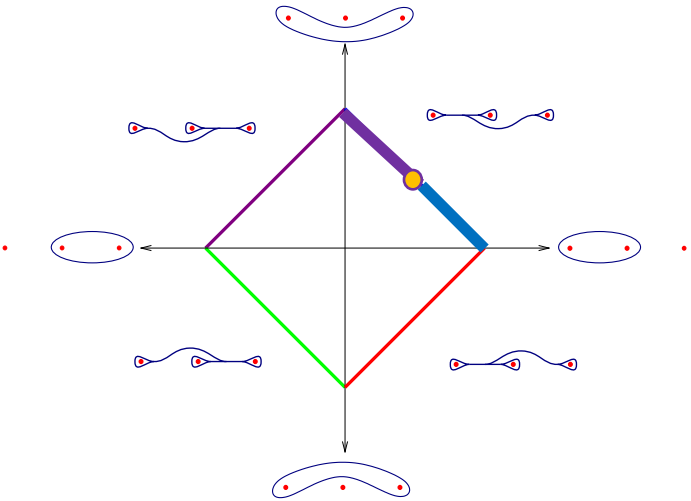


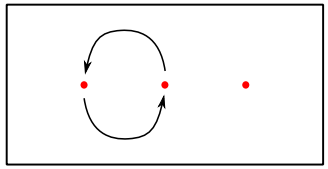


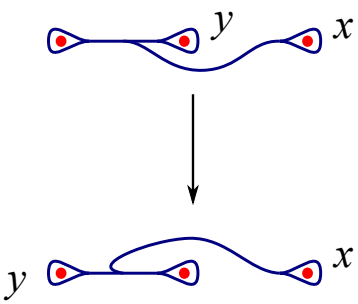


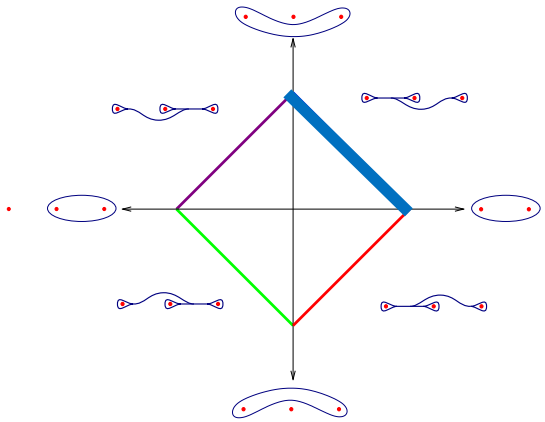


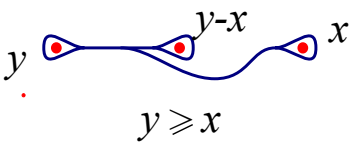
Natural splitting sequences

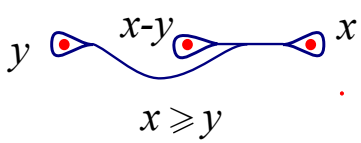


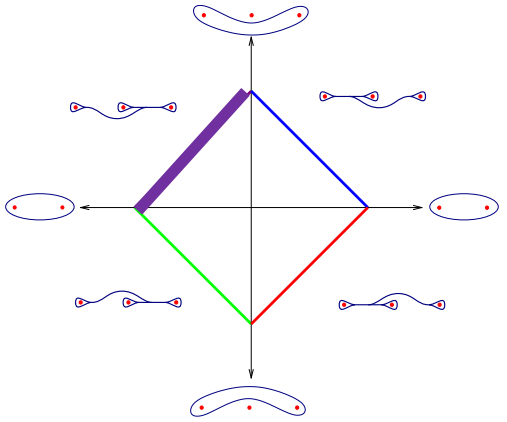












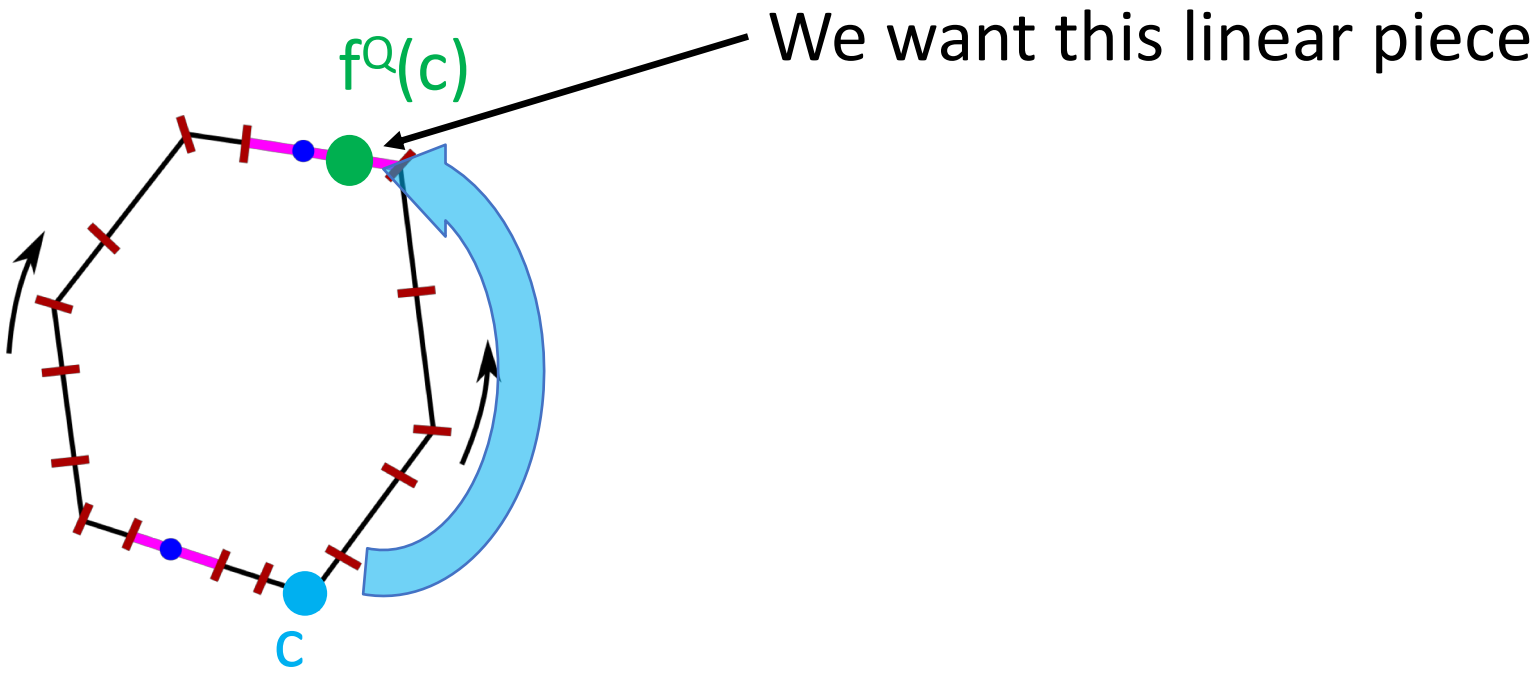
"Natural" splitting sequences associated to f:

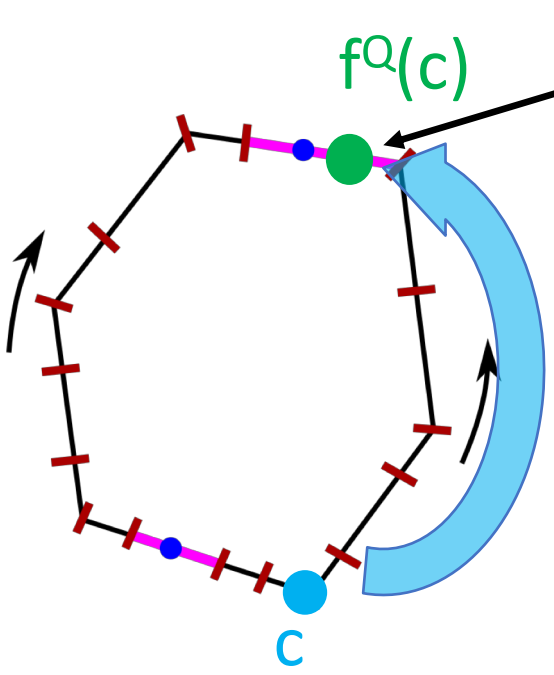
Apply the generators after each other and perform just as many splittings as necessary to be carried on a standard train track.

We get: $\tau_0 \to \cdots \to \tau_n$ *such that* $f(\tau_n)$ *is carried by a standard train track.*

Fact: n = O(N).

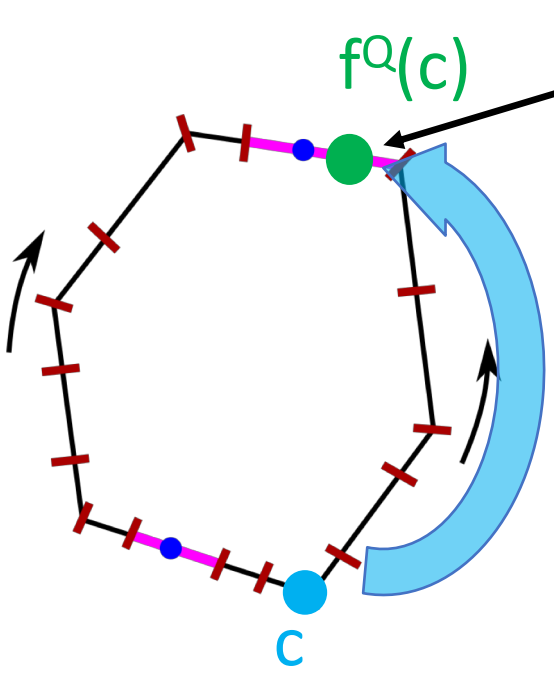
Problem: There are many such splitting sequences. Which one should we compute?


We want this linear piece
$f^Q(c)$
c



We want this linear piece

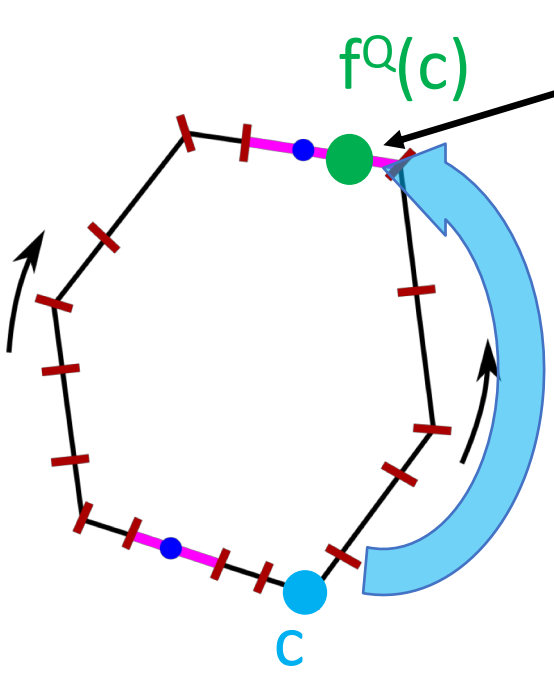
So we should always split such that F^u is carried.



We want this linear piece

So we should always split such that F^u is carried.

Problem: We don't know what F^u is.

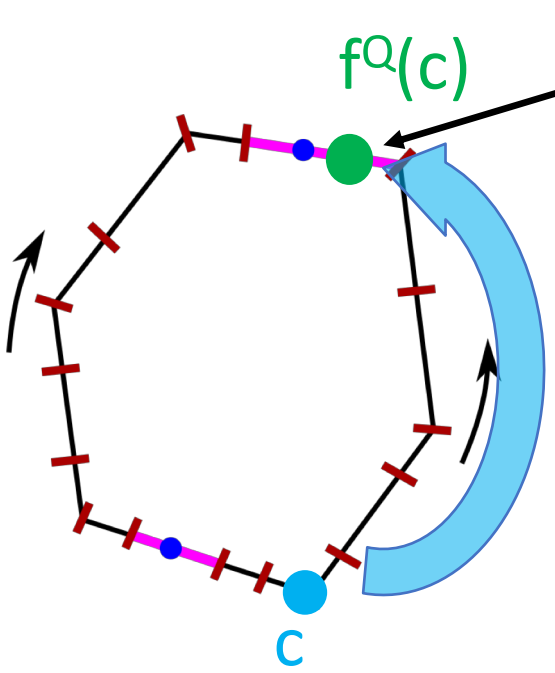


We want this linear piece

So we should always split such that F^u is carried.

Problem: We don't know what F^u is.

Solution: Split towards $f^Q(c)$ instead.



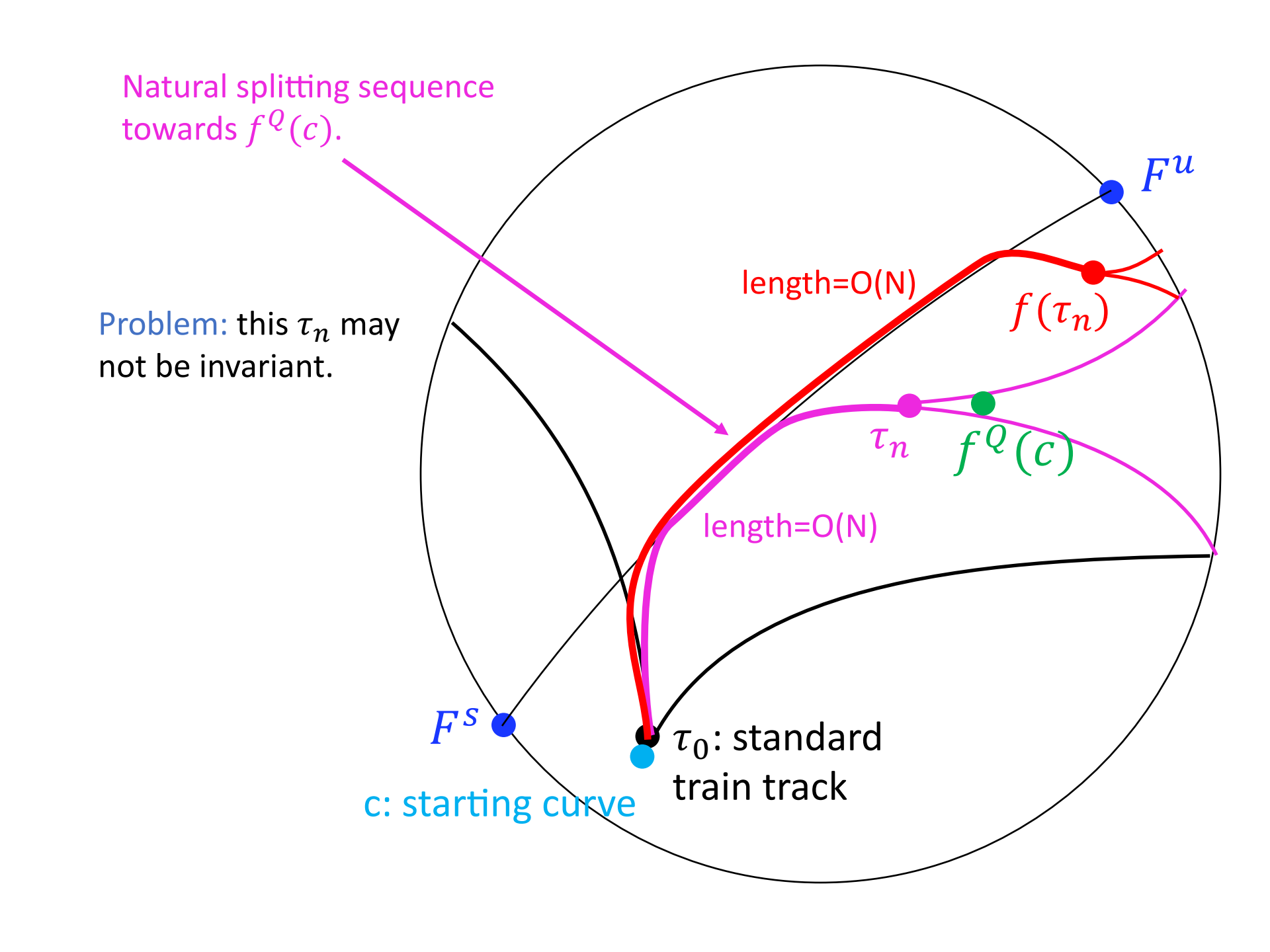
We want this linear piece

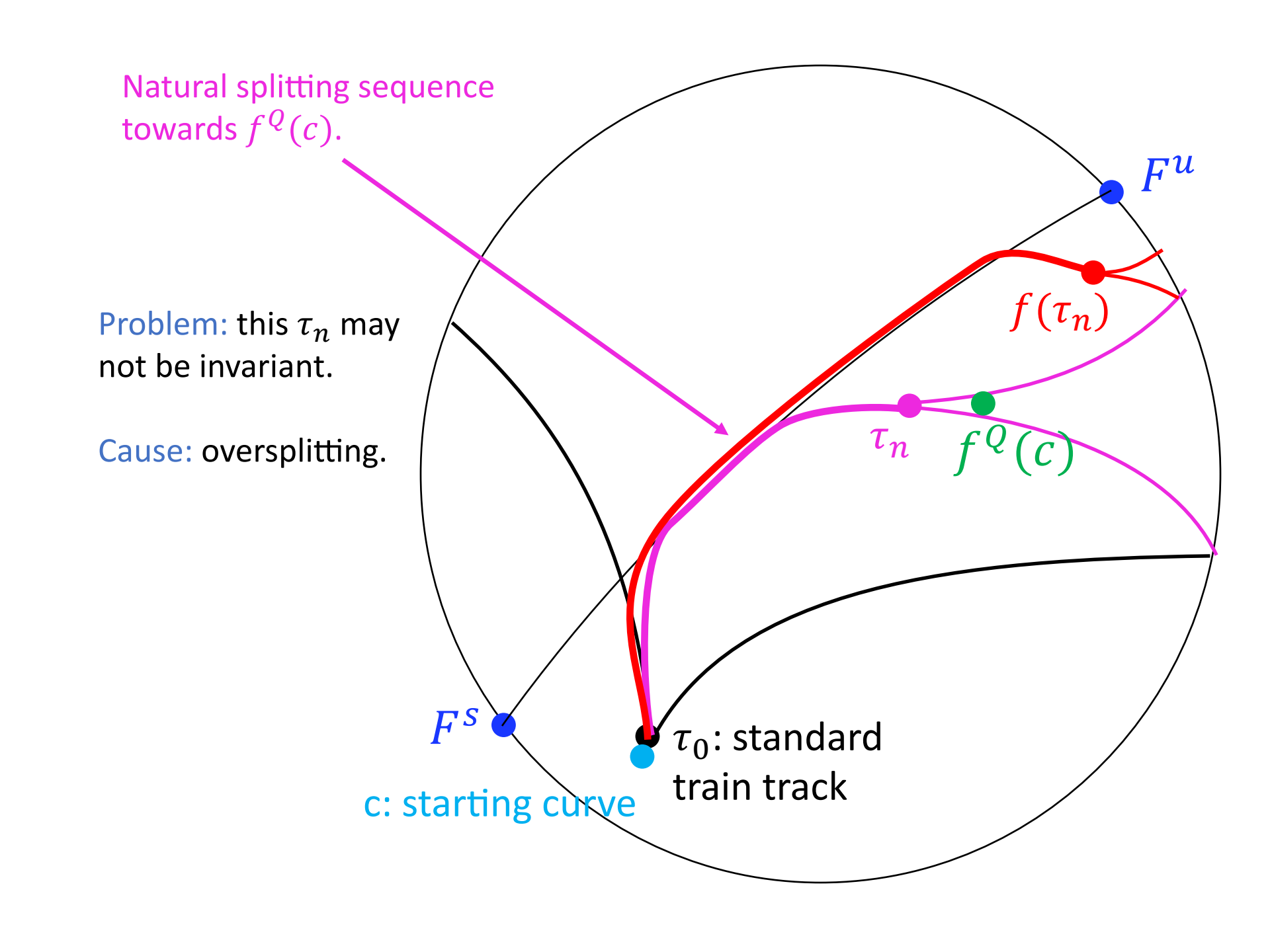
So we should always split such that F^u is carried.

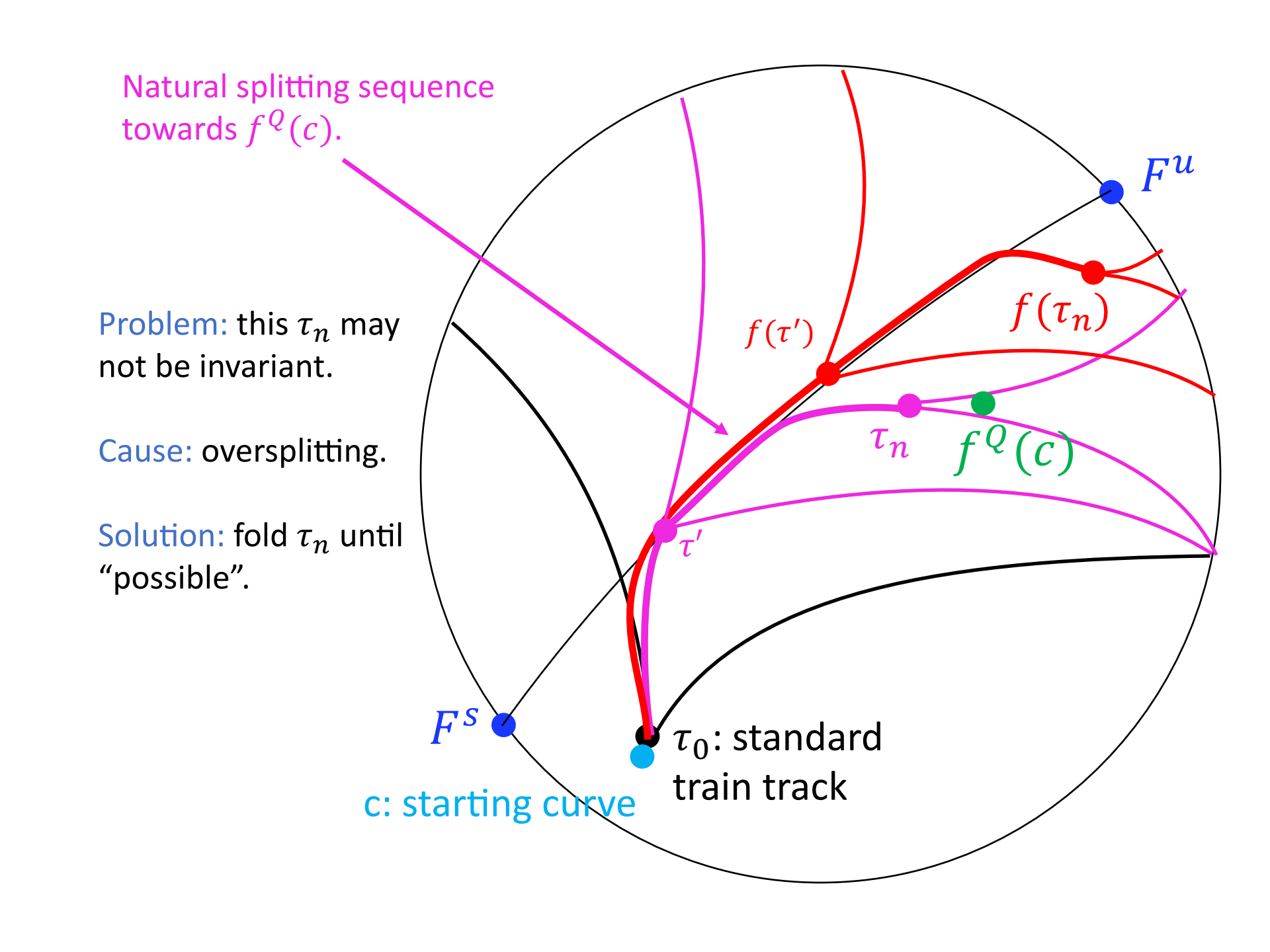
Problem: We don't know what F^u is.

Solution: Split towards $f^Q(c)$ instead.

We get: $\tau_0 \to \cdots \to \tau_n$ *such that* $f(\tau_n)$ *is carried by a standard train track and* $f^Q(c)$ is carried on τ_n.

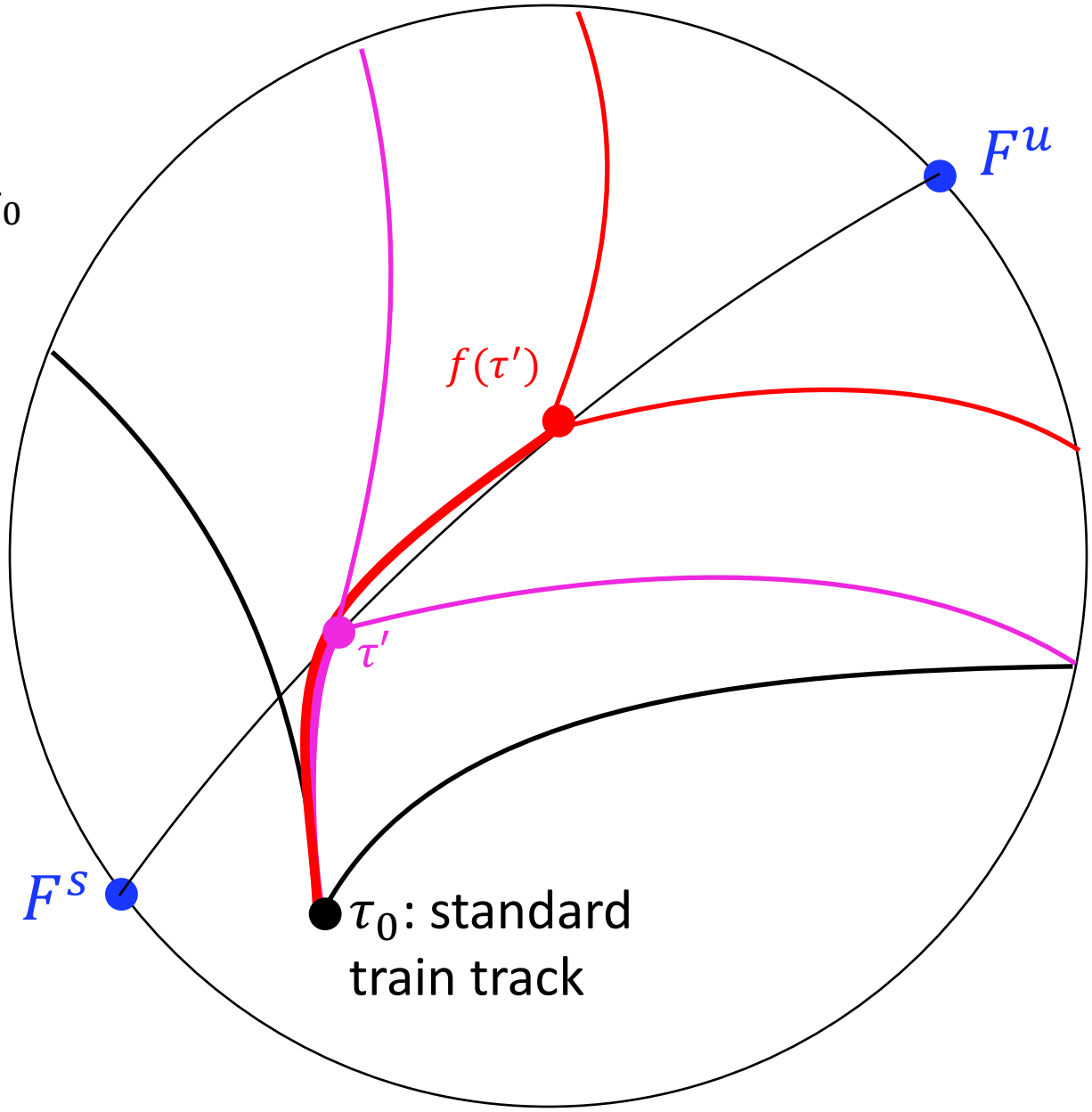

Natural splitting sequence towards $f^Q(c)$.
Problem: this τ_n may not be invariant.
F^u
length=O(N)
$f(\tau_n)$
τ_n
$f^Q(c)$
length=O(N)
F^s
τ_0: standard train track
c: starting curve


Natural splitting sequence towards $f^Q(c)$.
Problem: this τ_n may not be invariant.
Cause: oversplitting.
F^u
$f(\tau_n)$
τ_n
$f^Q(c)$
F^s
τ_0: standard train track
c: starting curve


Natural splitting sequence towards $f^Q(c)$.
Problem: this τ_n may not be invariant.
Cause: oversplitting.
Solution: fold τ_n until "possible".
F^u
$f(\tau')$
$f(\tau_n)$
τ_n
$f^Q(c)$
τ'
F^s
τ_0: standard train track
c: starting curve

We have:

- τ' carried on τ_0
- $f(\tau')$ carried on τ_0
- we cannot fold τ' anymore

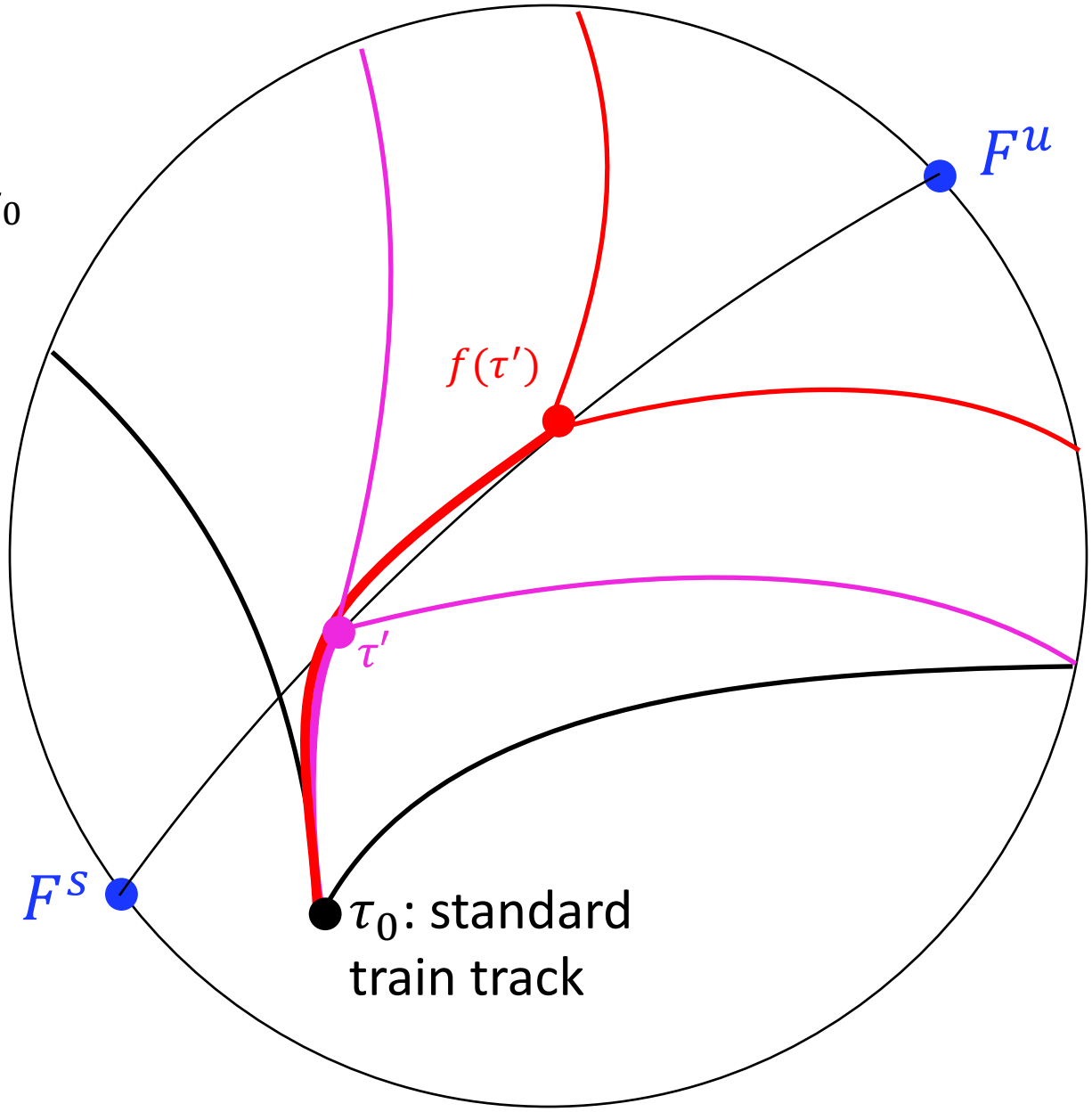


We have:

- τ' carried on τ_0
- $f(\tau')$ carried on τ_0
- we cannot fold τ' anymore

Morally true proposition:

If f is pA, then $f(\tau')$ is carried on τ'. In other words, τ' is an invariant train track.

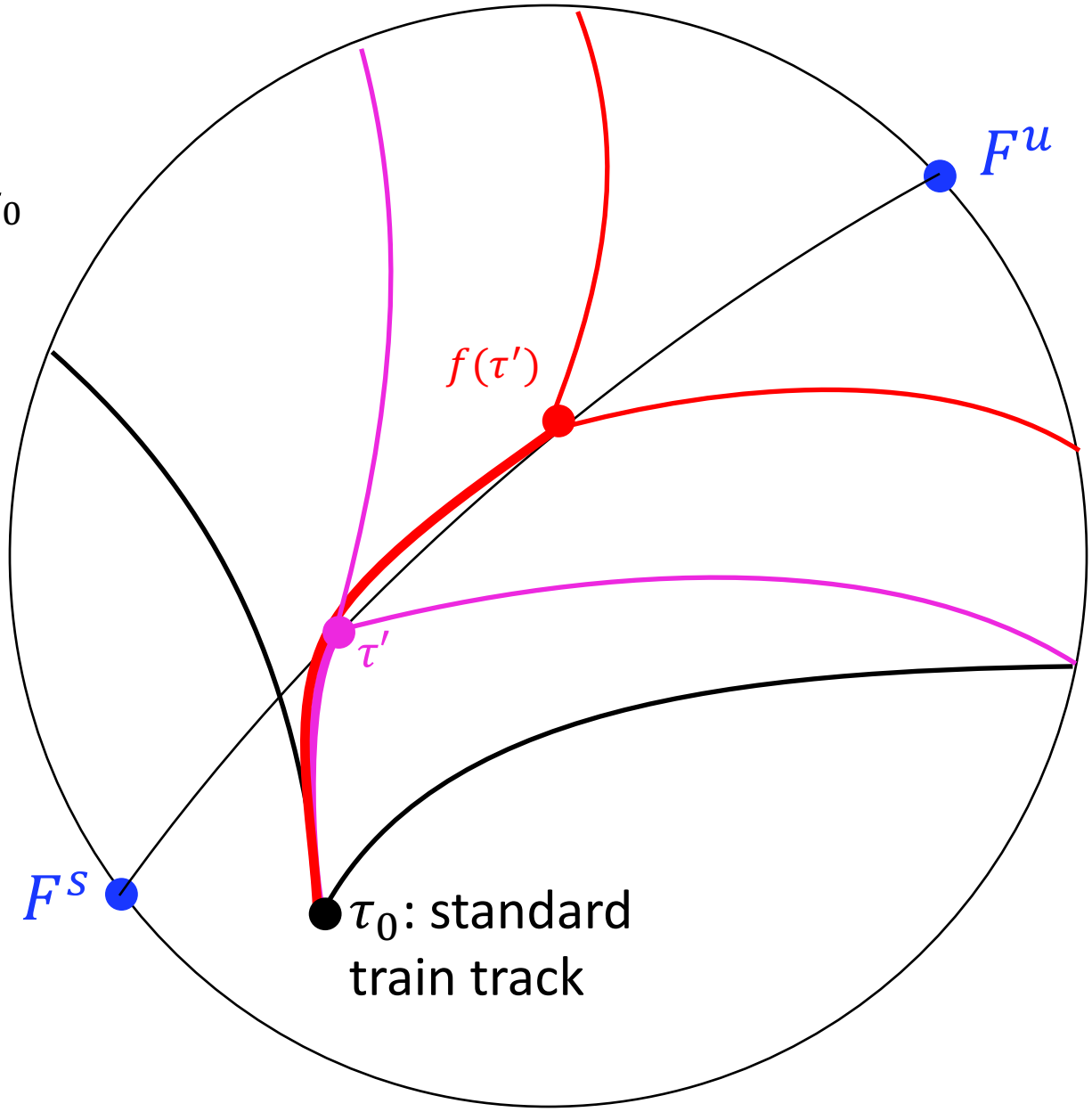


We have:

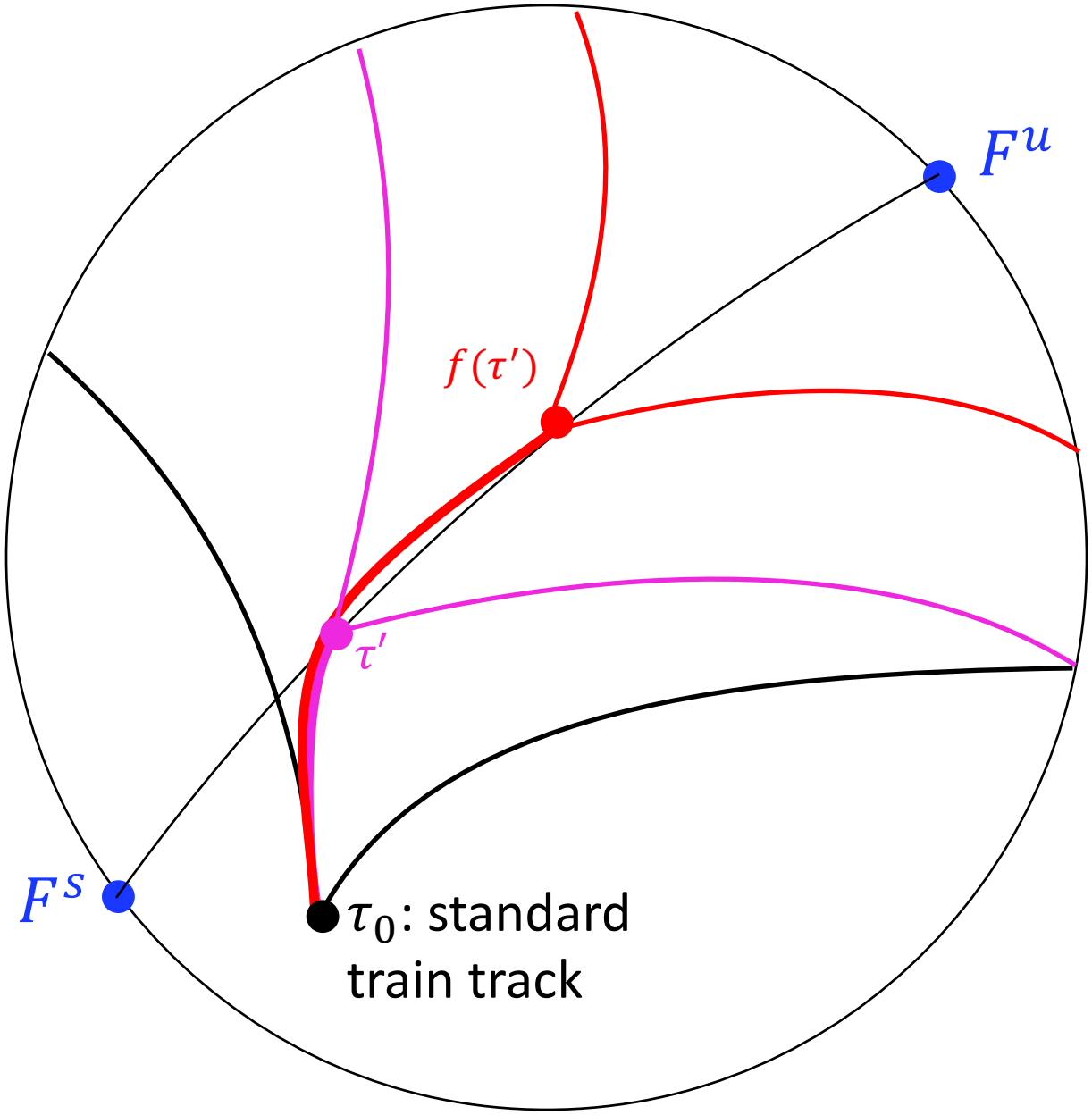
- τ' carried on τ_0
- $f(\tau')$ carried on τ_0
- we cannot fold τ' anymore

Morally true proposition:
If f is pA, then $f(\tau')$ is carried on τ'. In other words, τ' is an invariant train track.

If f is reducible, then the twisting reducible curves and invariant train tracks for partial pseudo-Anosovs appear as subtracks of τ'.

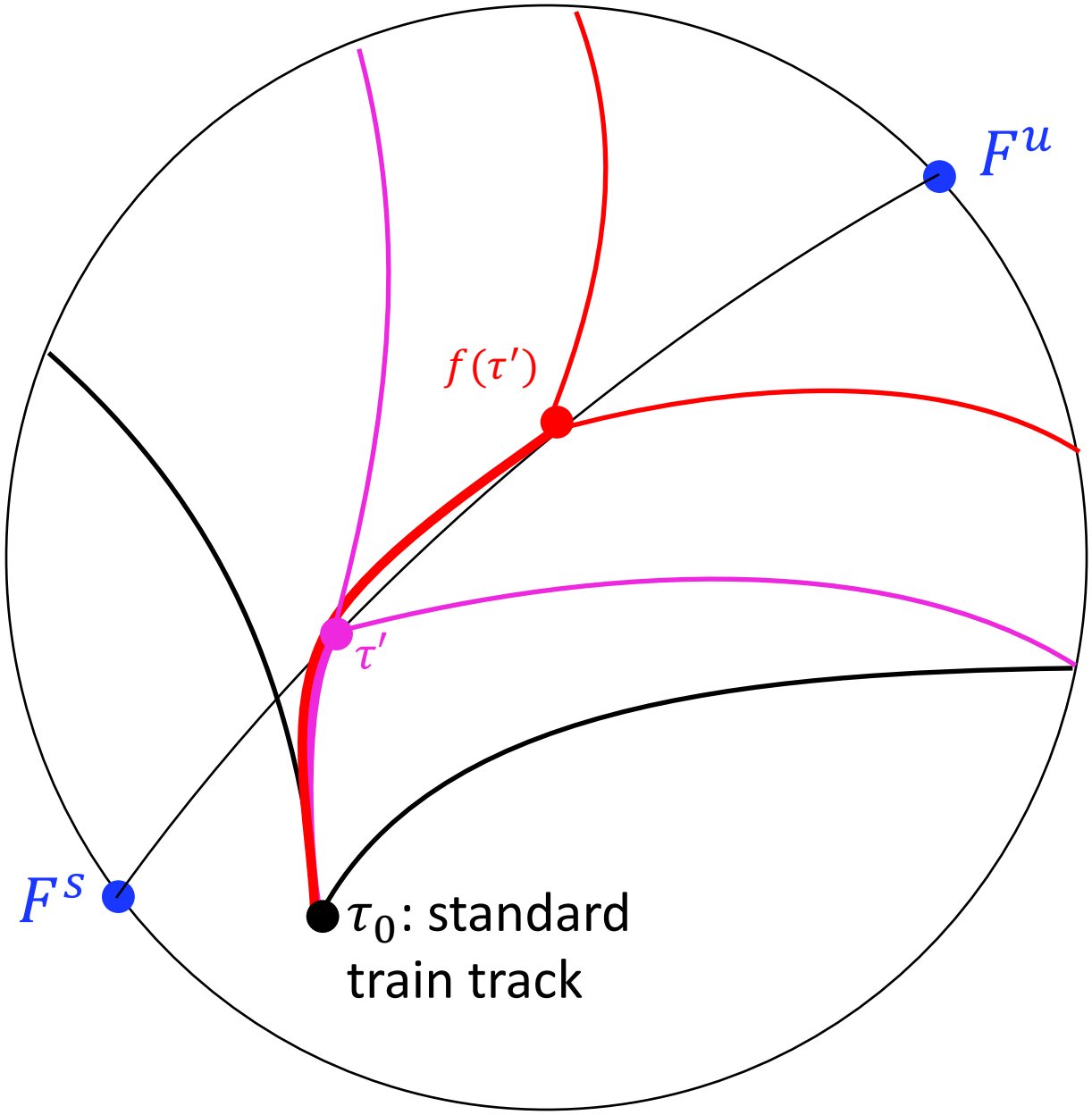


We still need to compute how $f(\tau')$ is carried on τ'.



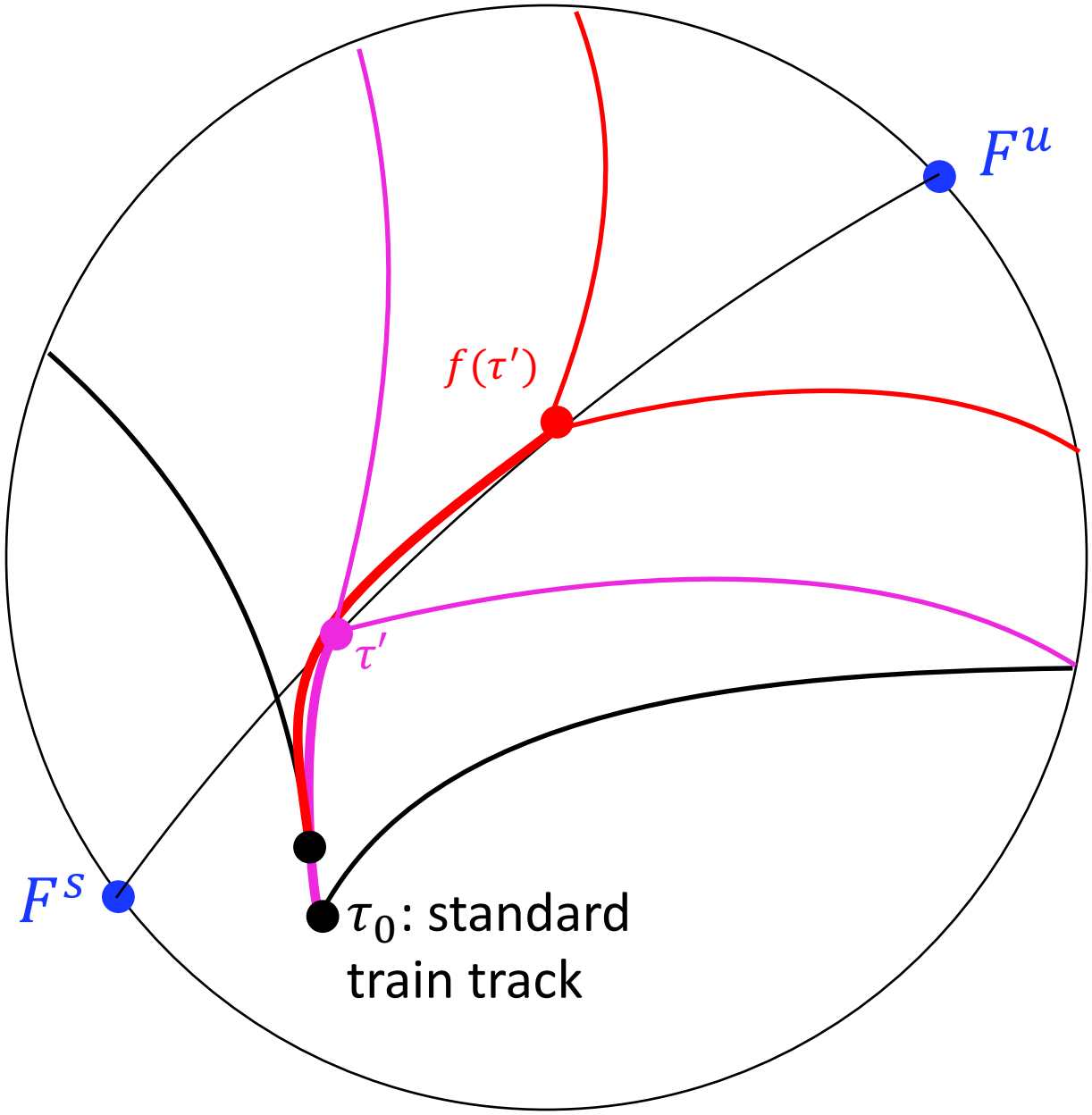
We still need to compute how $f(\tau')$ is carried on τ'.

Solution:
Split τ_0 towards τ' and update how $f(\tau')$ is carried.



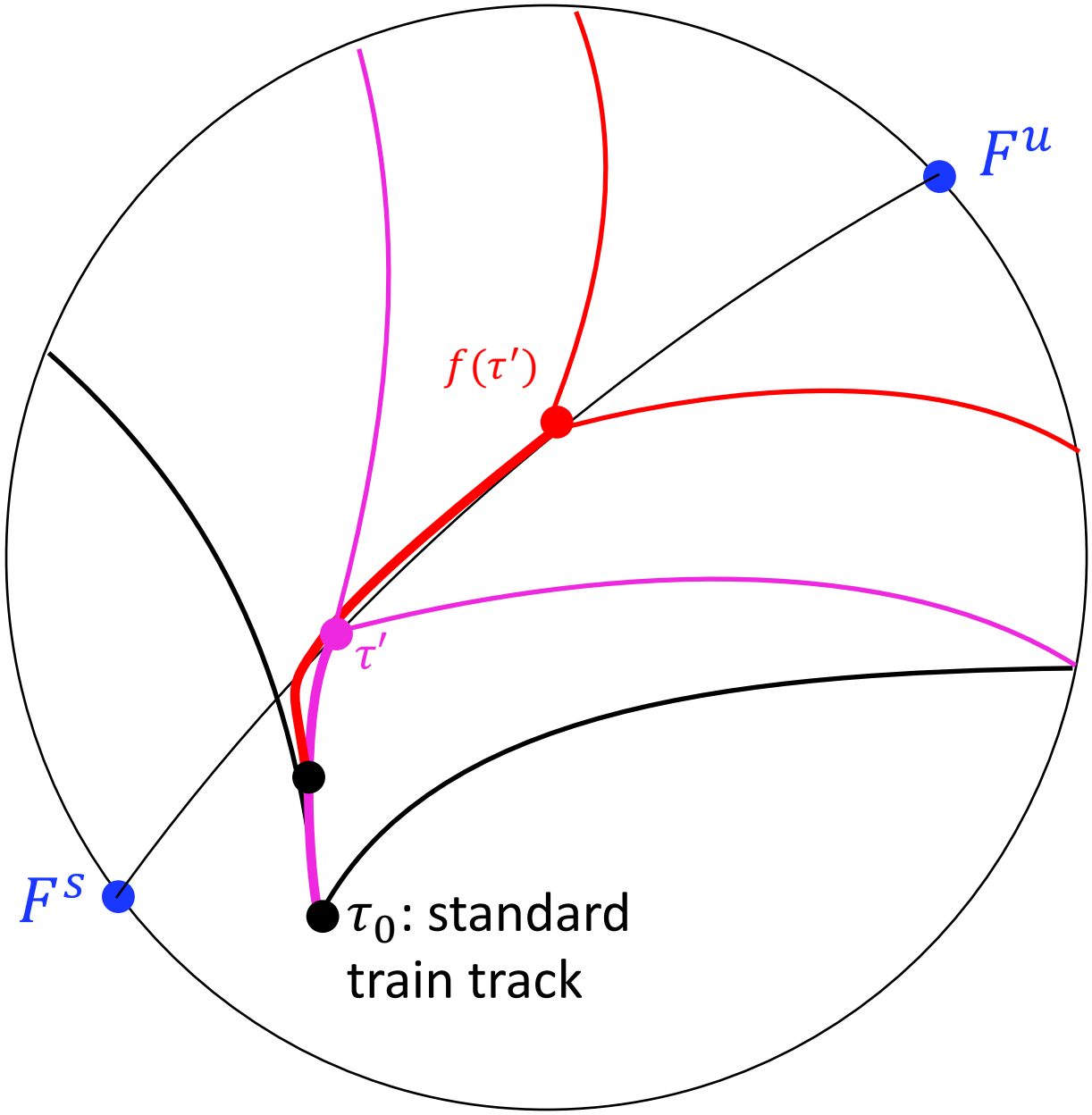
We still need to compute how $f(\tau')$ is carried on τ'.

Solution:
Split τ_0 towards τ' and update how $f(\tau')$ is carried.



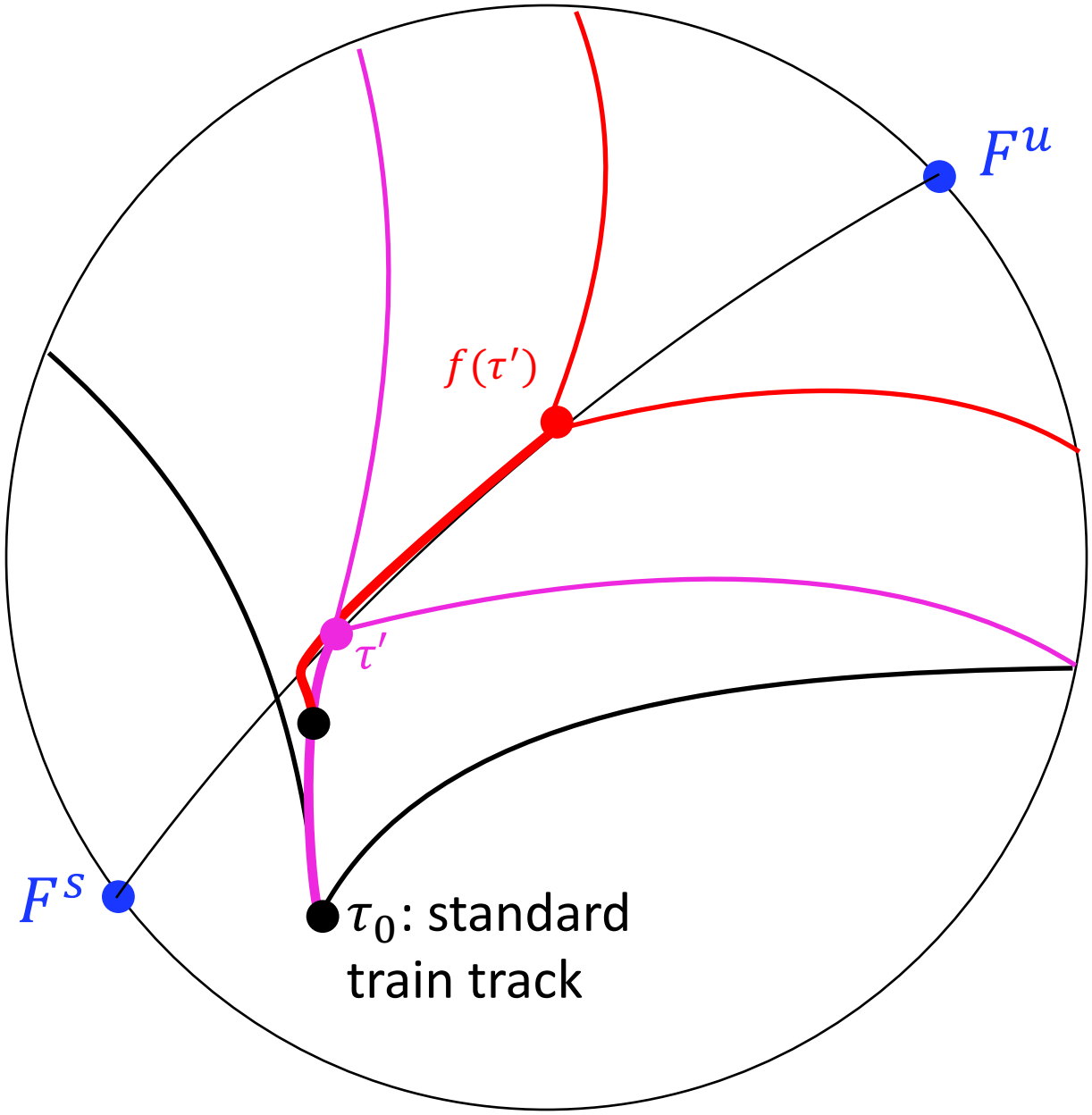
We still need to compute how $f(\tau')$ is carried on τ'.

Solution:
Split τ_0 towards τ' and update how $f(\tau')$ is carried.



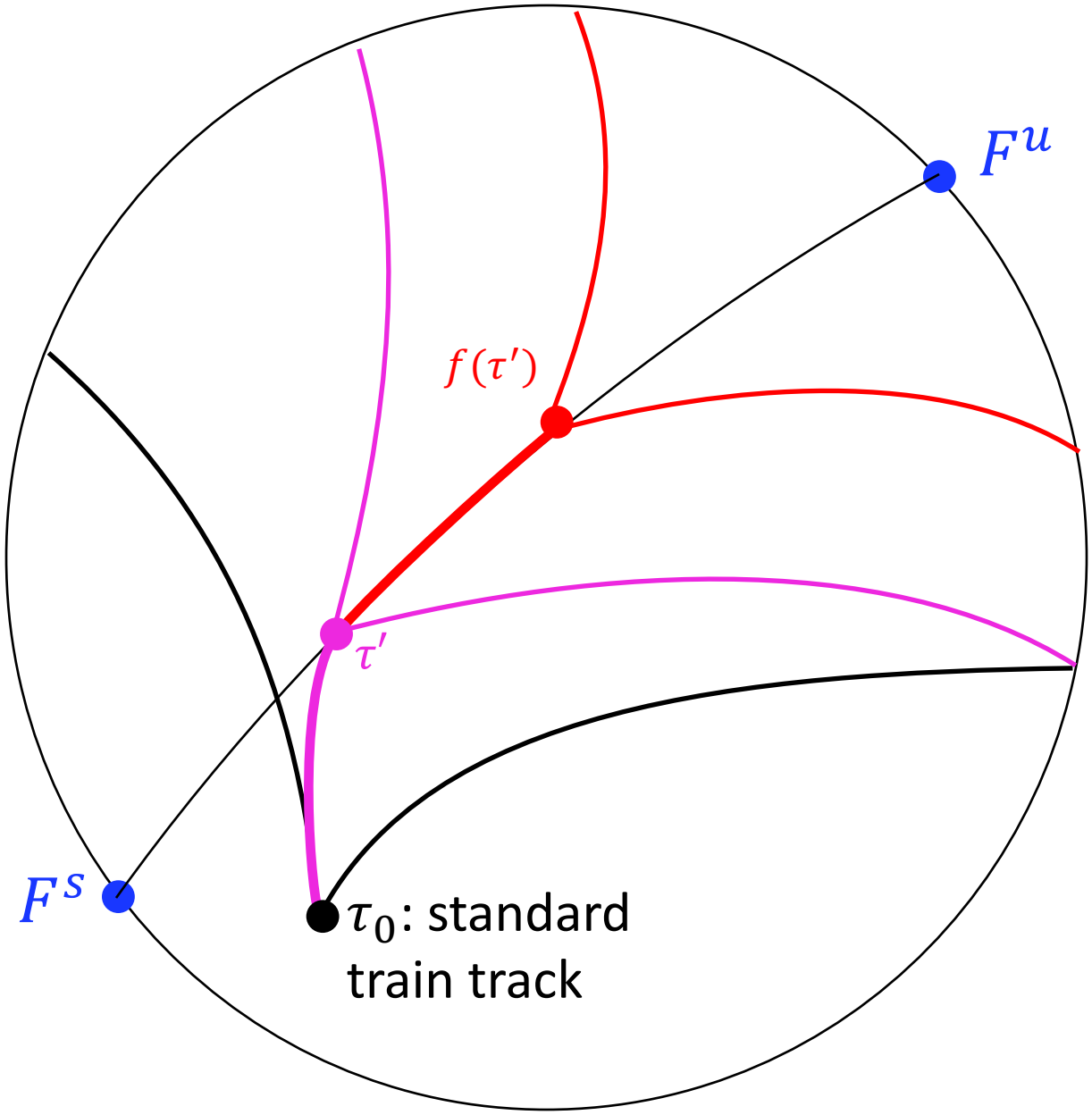
We still need to compute how $f(\tau')$ is carried on τ'.

Solution:
Split τ_0 towards τ' and update how $f(\tau')$ is carried.



We still need to compute how $f(\tau')$ is carried on τ'.

Solution:
Split τ_0 towards τ' and update how $f(\tau')$ is carried.



Improvements over v2

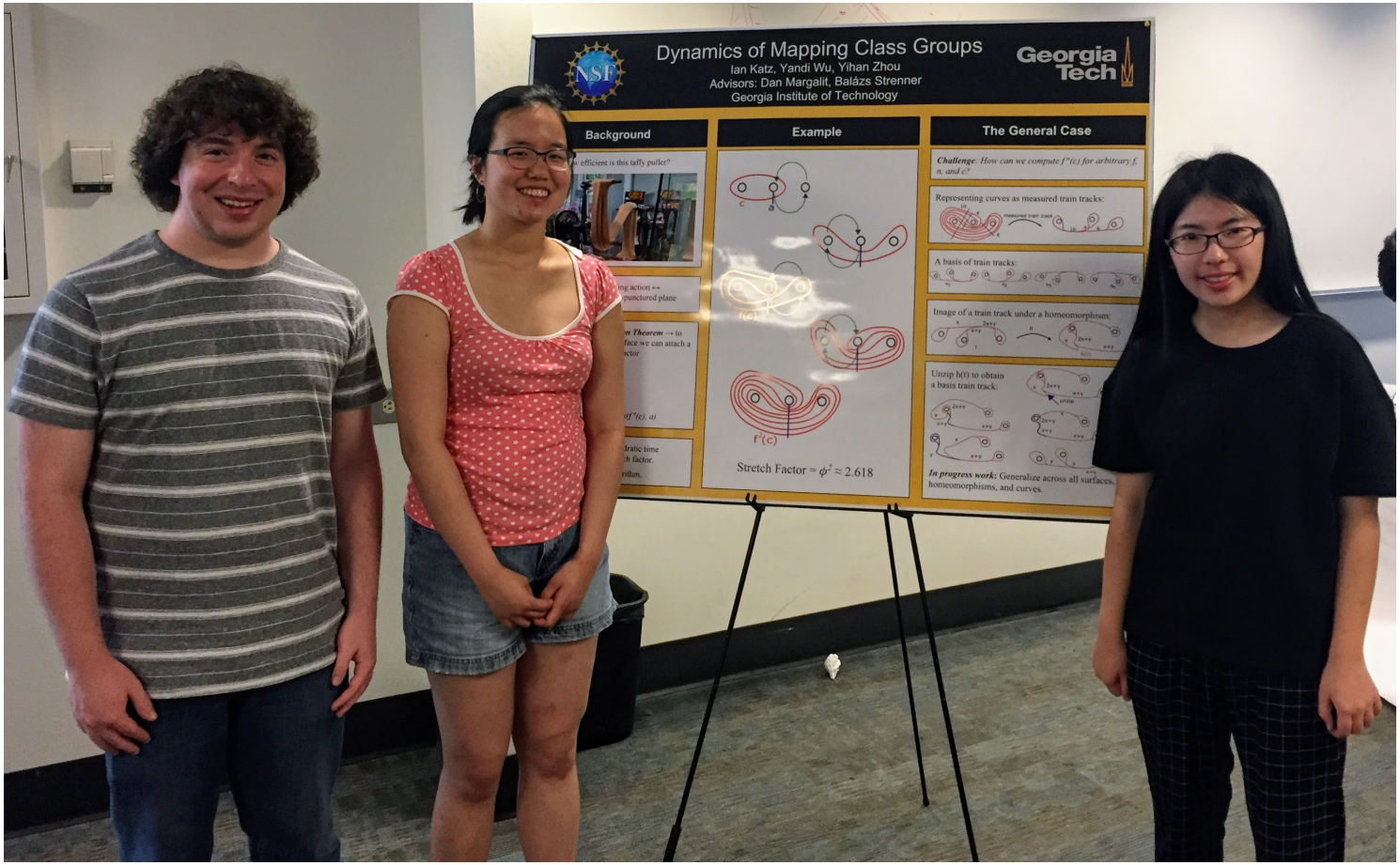
- A natural, more geometric way to obtain the reducing curve, invariant train tracks.
- No dependence on polynomial factorization algorithms.
- Better bounds on the length of the splitting sequence.
- No PMF, just train tracks. More hope to adapt to $\mathrm{Out}(F_n)$.

Main ingredients

- Characterizations (using slope) for a train track carrying a curve or another train track.
- Efficient representation of a train track carrying another train track.
- Show that various operations (splitting/folding the carrying/carried track) can be performed efficiently.
- Work out how those ”natural” splittings work on closed surfaces with pants decomposition markings.

Macaw
(implementation)

REU 2017 (Georgia Tech)

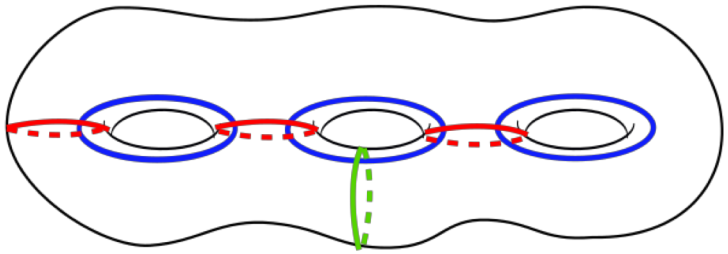


Ian Katz Yandi Wu Yihan Zhou

1. Works for closed surfaces

2. Solves the word problem/checks relations

```
sage: A,B,c = humphries_generators(3)
sage: A[0]*B[0] == B[0]*A[0]
False
sage: A[0]*B[0]*A[0] == B[0]*A[0]*B[0]
True
```



3. Computes the order

```
sage: g = hyperelliptic_involution(4)
sage: g.order()
2
```

Thank you!