Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

# Bundled Crossings Revisited

**Steven Chaplick**, Thomas C. van Dijk, Myroslav Kryven,
Alexander Wolff

Julius-Maximilians-Universität Würzburg, Germany

Ji-won Park

KAIST, Daejeon, Republic of Korea

Alexander Ravsky

Pidstryhach Institute for Applied Problems of Mechanics and Mathematics,

Nat. Acad. Sciences of Ukraine, Lviv, Ukraine

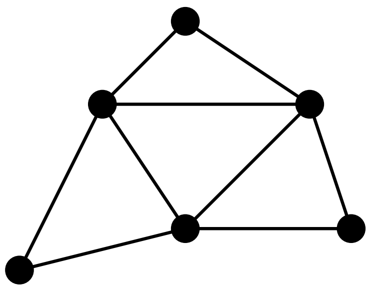
# Motivation

Ideally drawings of graphs should avoid crossings ...

# Motivation

Ideally drawings of graphs should avoid crossings ...

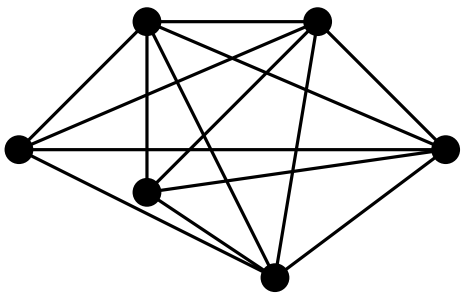
Planar graphs can be drawn without crossings



# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

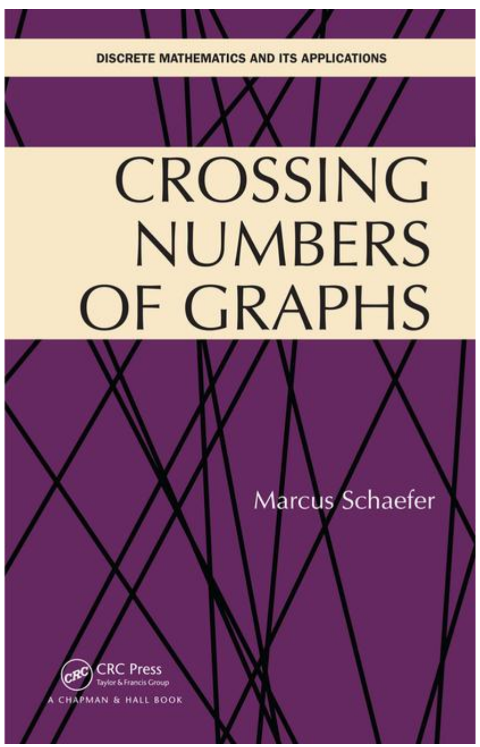


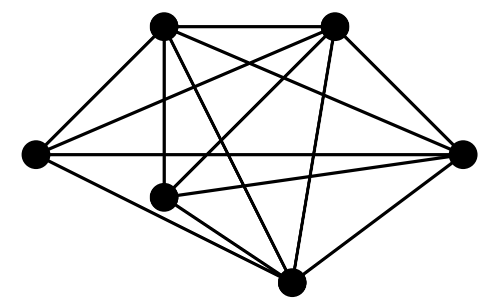
# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?



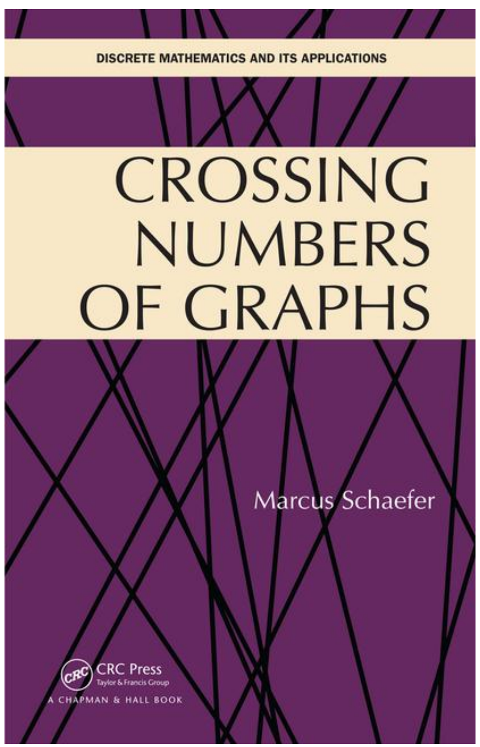



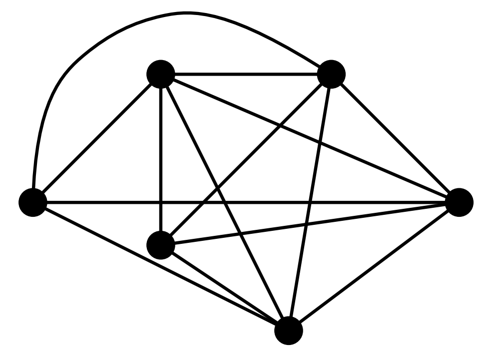
# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?



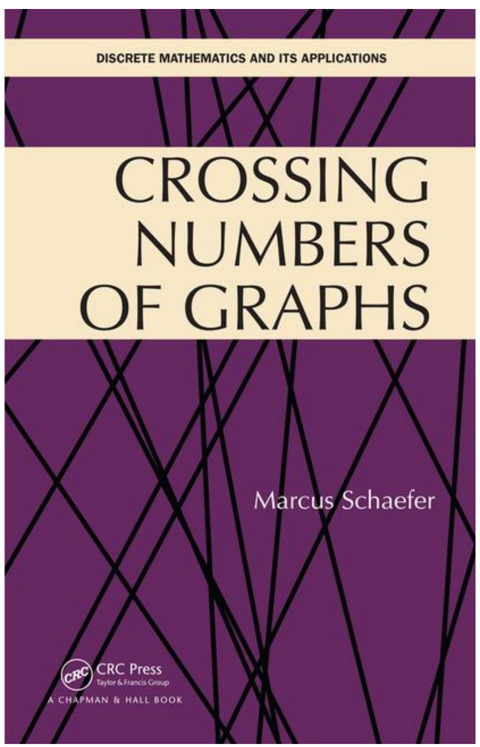
# Motivation

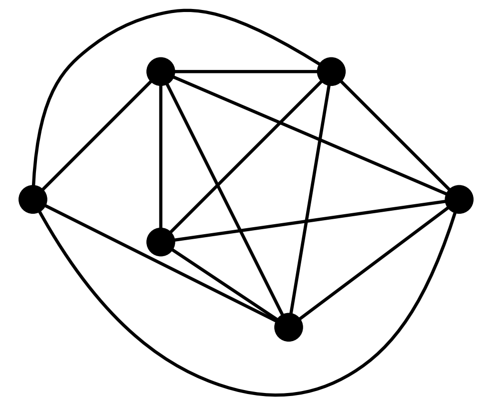
Ideally drawings of graphs should avoid crossings ...

Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?



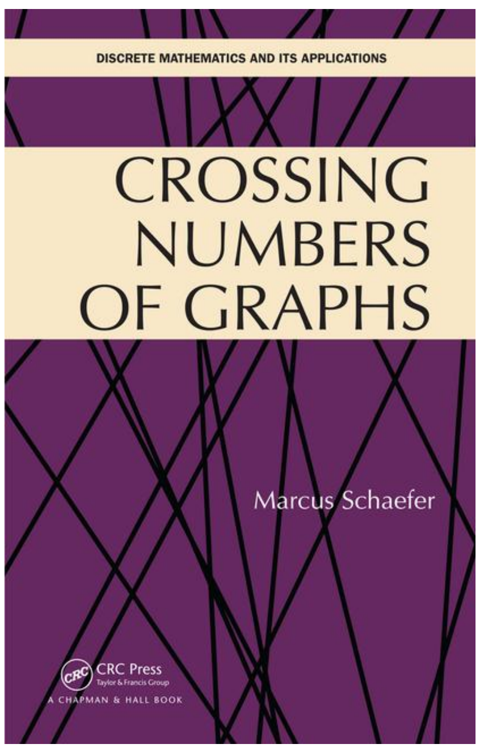



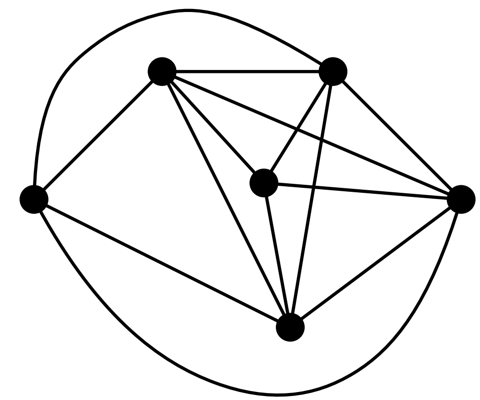
# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?



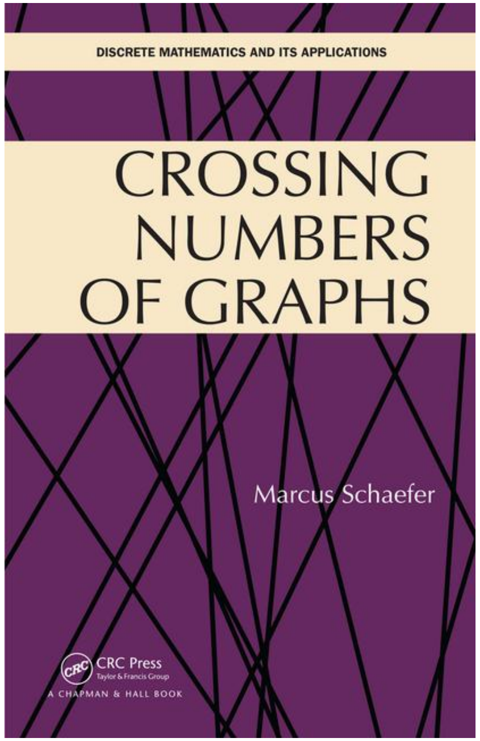



# Motivation

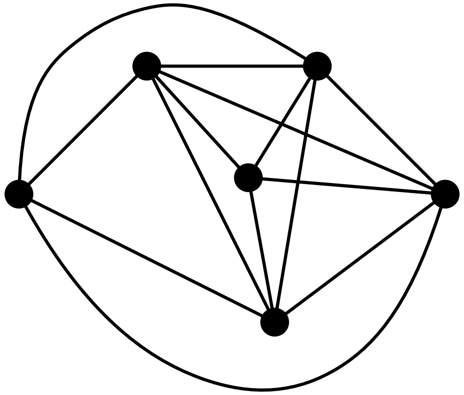
Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?




Lots of different variants.

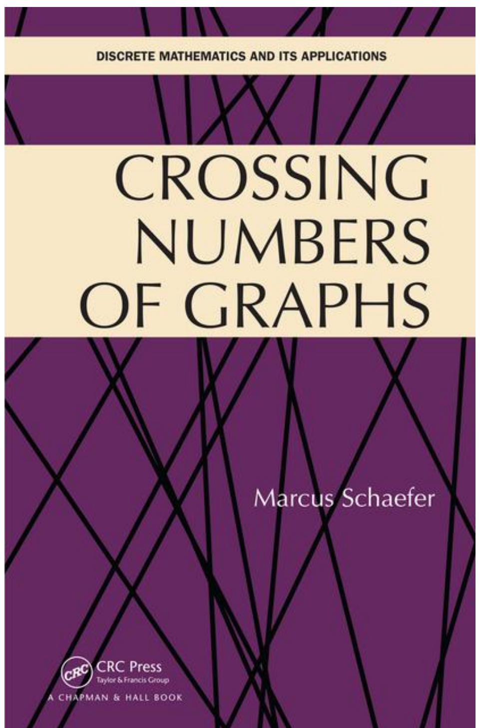


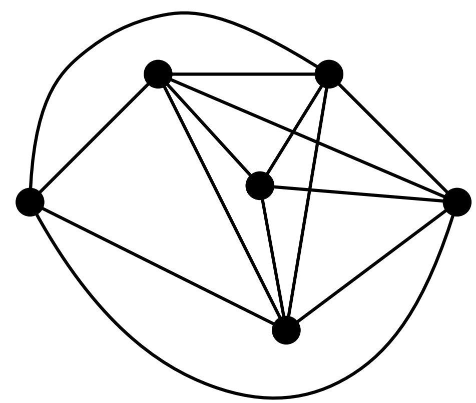
# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

Classical problem in Graph Drawing:
How to minimize the number of crossings?

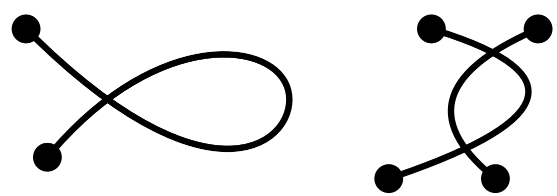


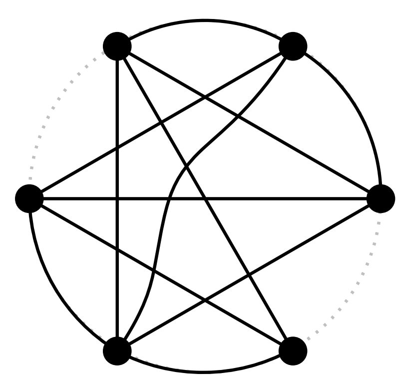



Lots of different variants.

Our main result concerns simple circular layouts.

simple avoids:



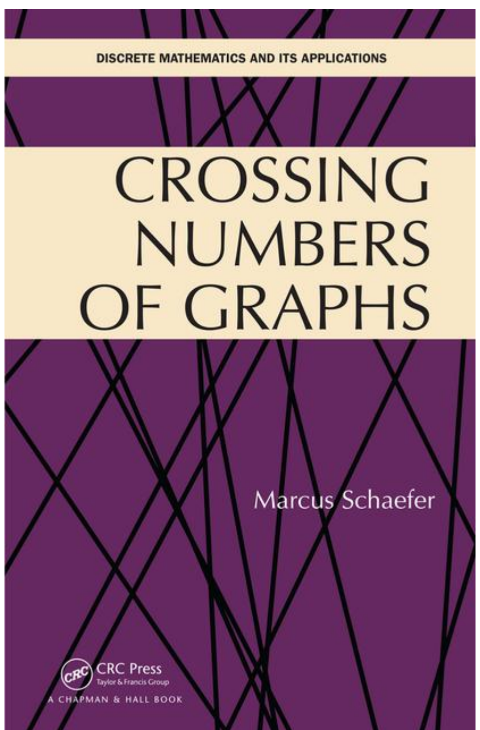


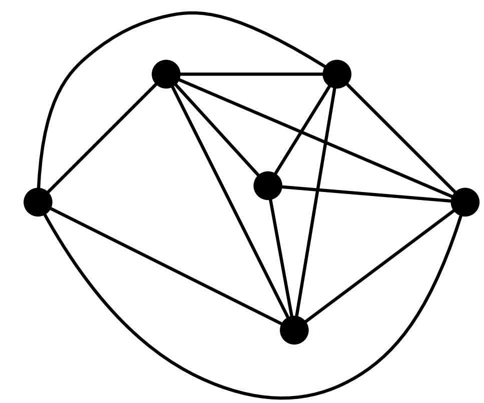
# Motivation

Ideally drawings of graphs should avoid crossings ...
Planar graphs can be drawn without crossings

but many graphs cannot be drawn without crossings.

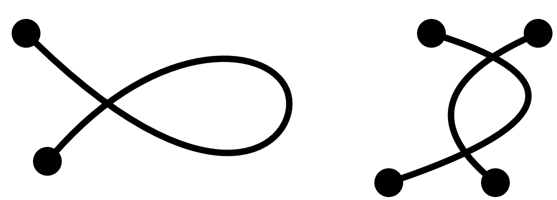
Classical problem in Graph Drawing:
How to minimize the number of crossings?

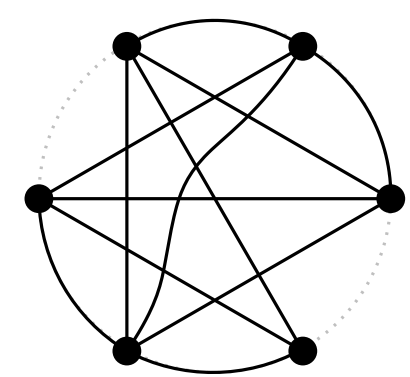


Lots of different variants.

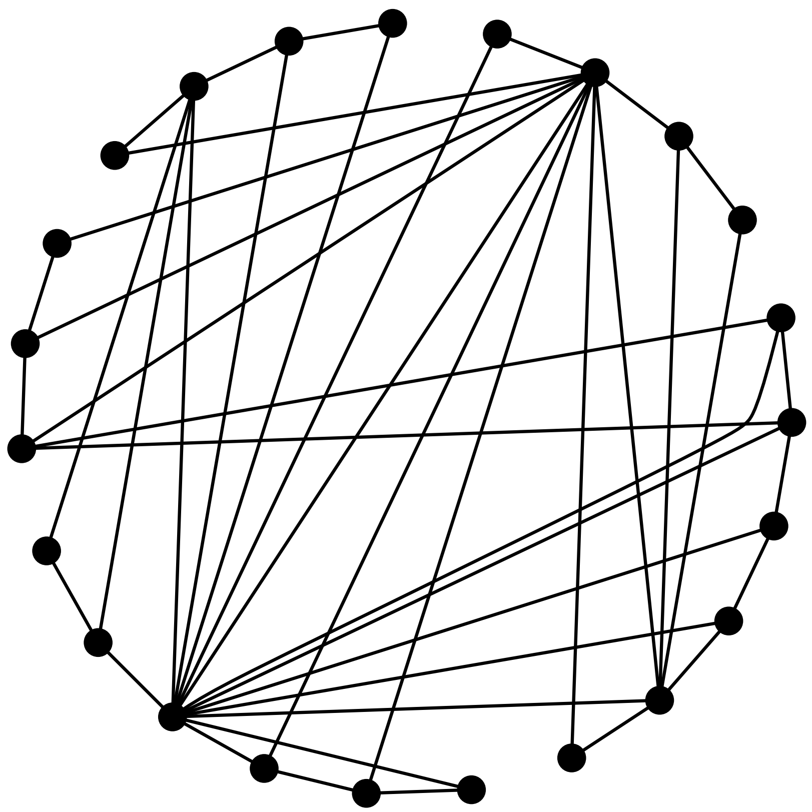
Our main result concerns simple circular layouts.

simple avoids: 



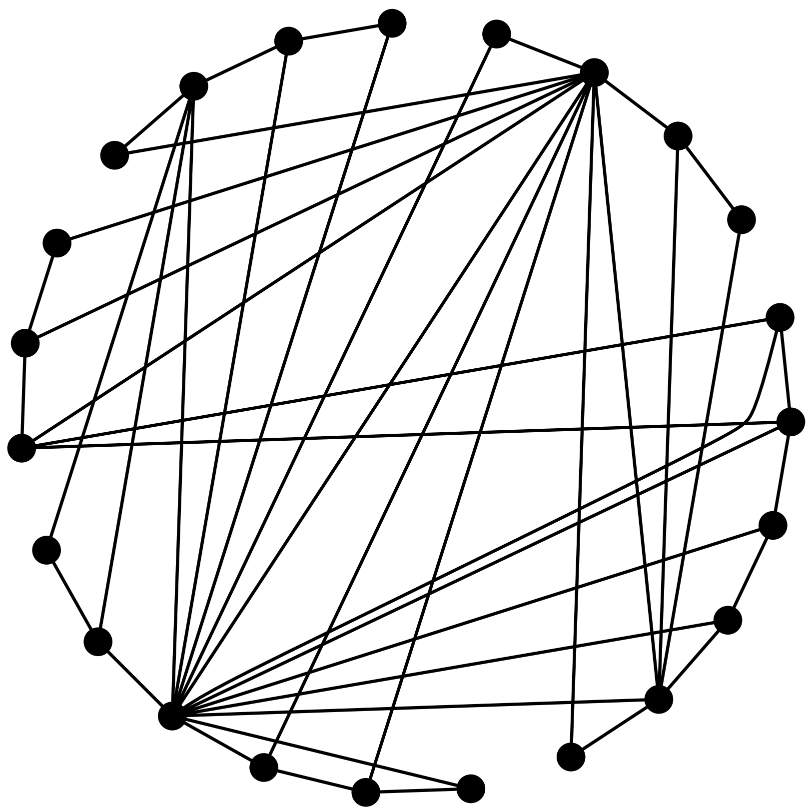
This talk concerns *bundled crossings*, def'd next.

# Motivation



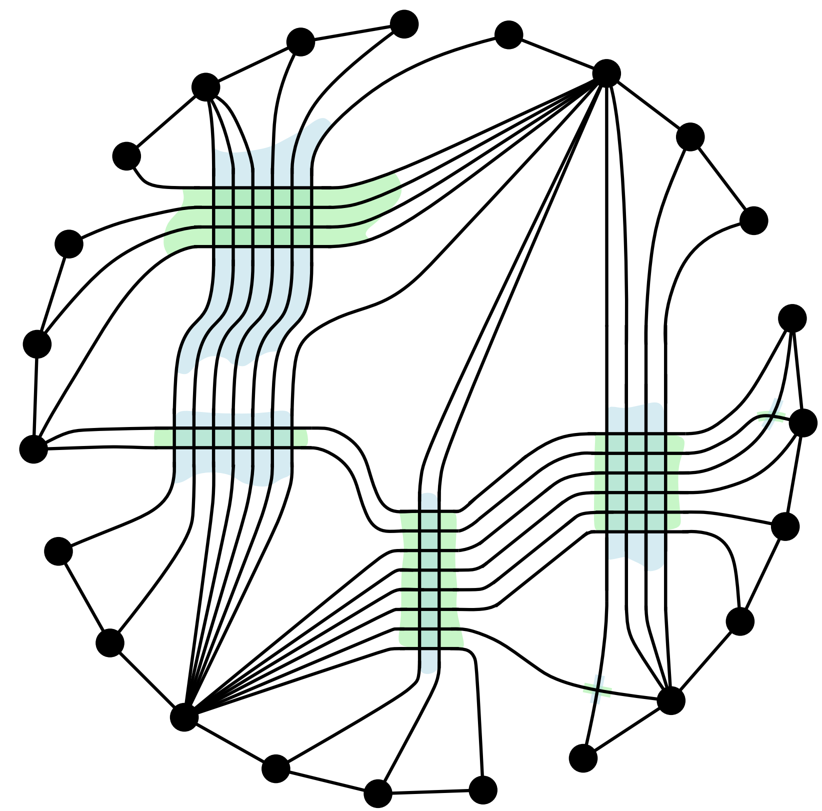
There is an FPT algorithm for deciding whether a graph admits a circular layout with $k$ crossings. [Bannister, Eppstein '14]

# Motivation



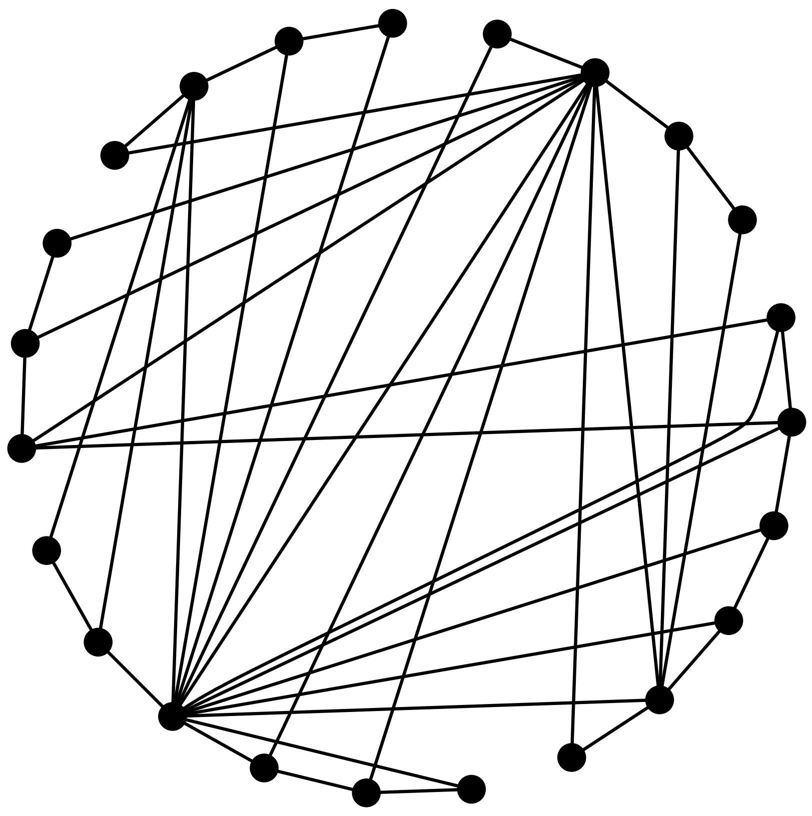
[Holten '06]

Bundle the drawing



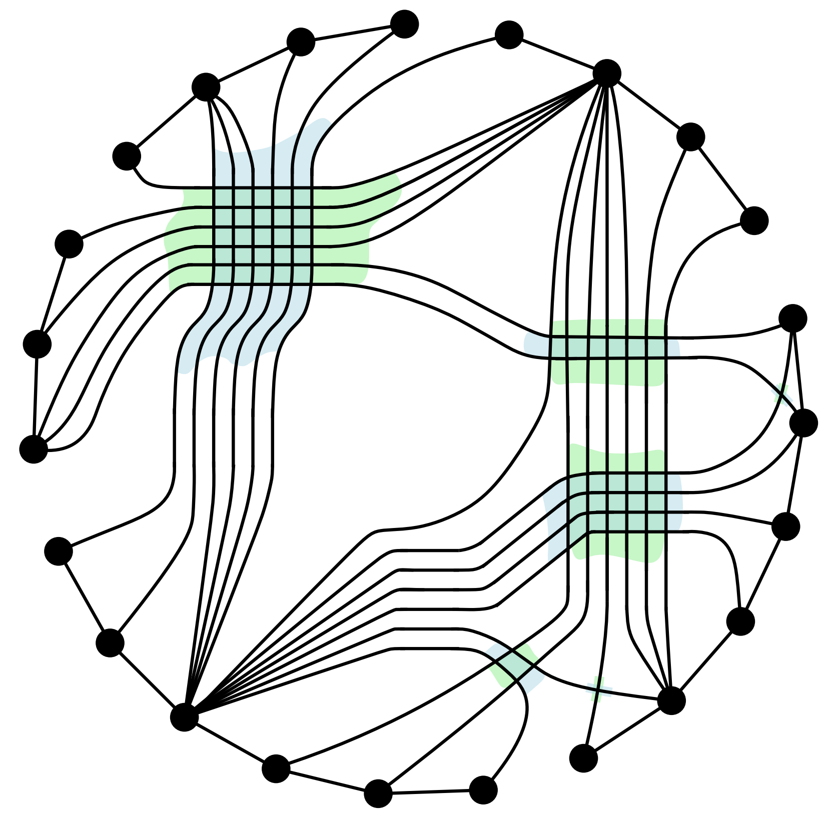
There is an FPT algorithm for deciding whether a graph admits a circular layout with $k$ crossings.[Bannister, Eppstein '14]

# Motivation



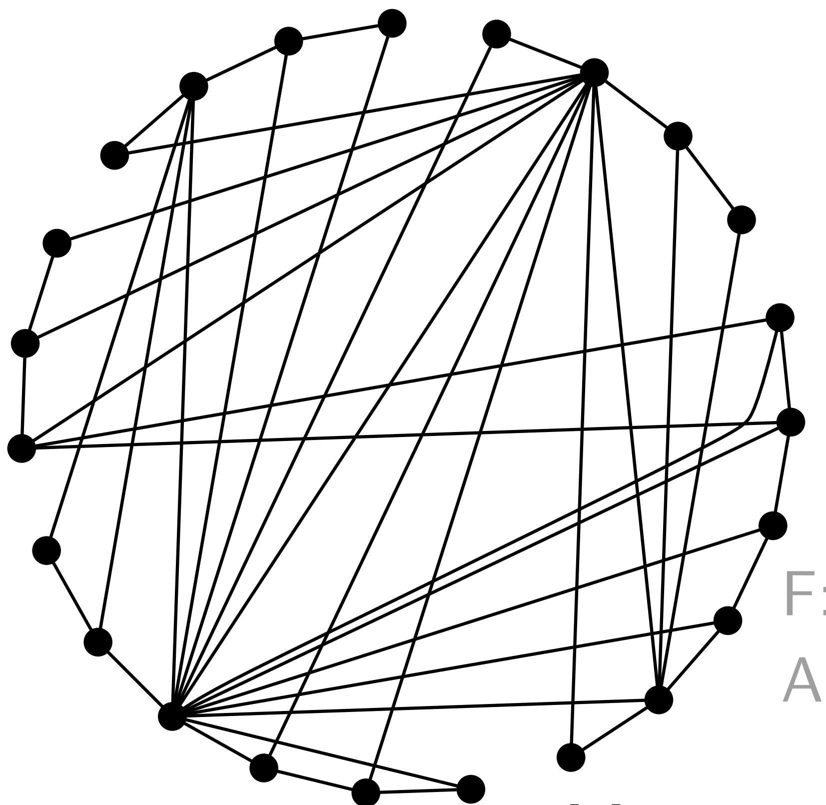
[Holten '06]

Bundle the drawing



There is an FPT algorithm for deciding whether a graph admits a circular layout with $k$ crossings.[Bannister, Eppstein '14]

# Motivation

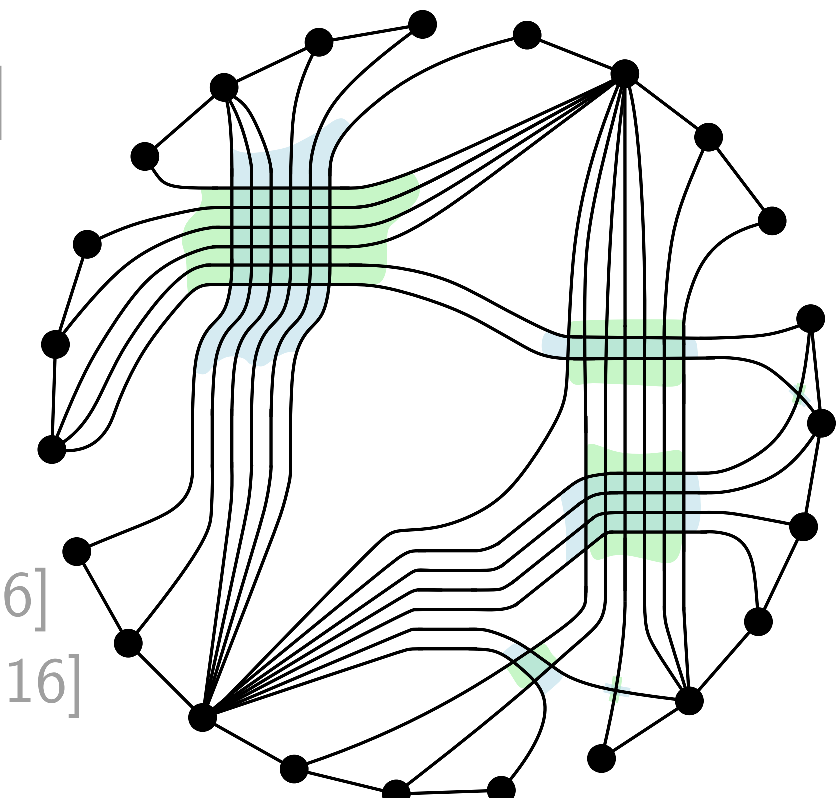


[Holten '06]

Bundle the drawing

F: [Fink et al. '16]

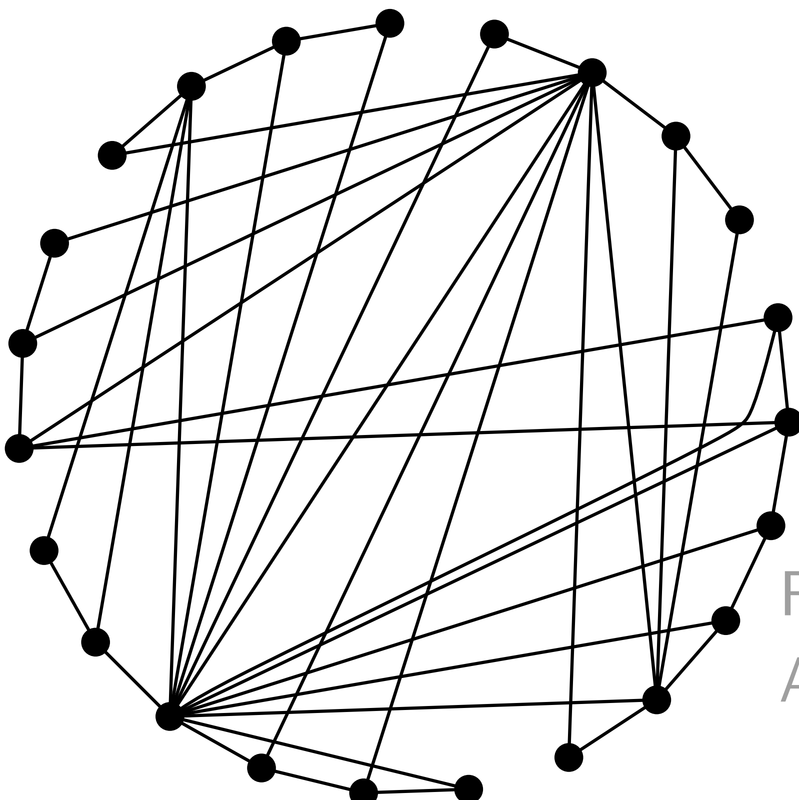
A: [Alam et al. '16]



Minimize crossings of bundles instead of edges!

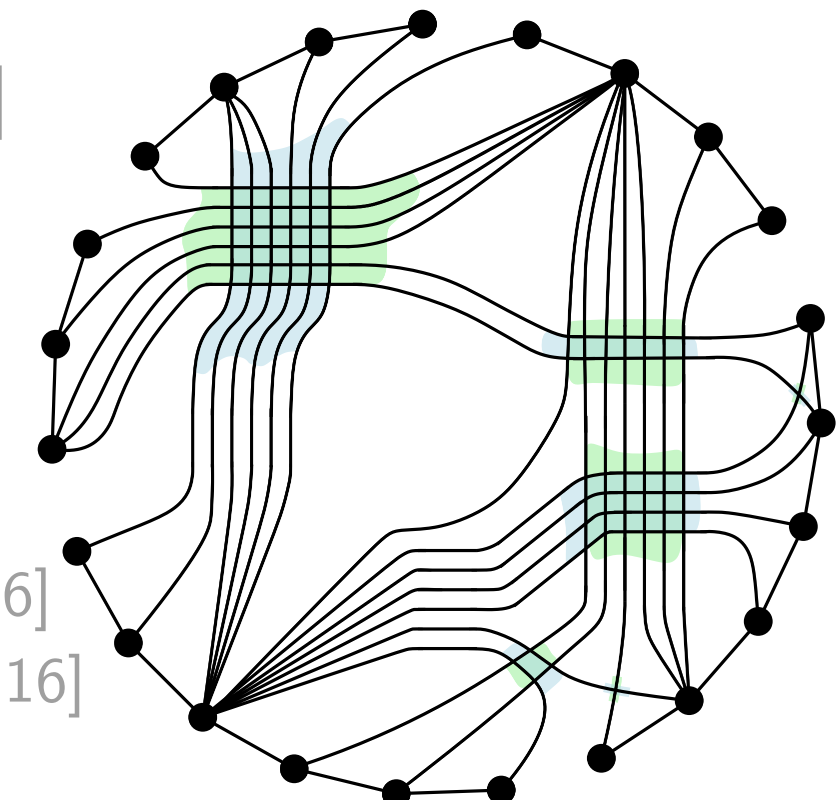
There is an FPT algorithm for deciding whether a graph admits a circular layout with $k$ crossings.[Bannister, Eppstein '14]

# Motivation



[Holten '06]

Bundle the drawing



F: [Fink et al. '16]

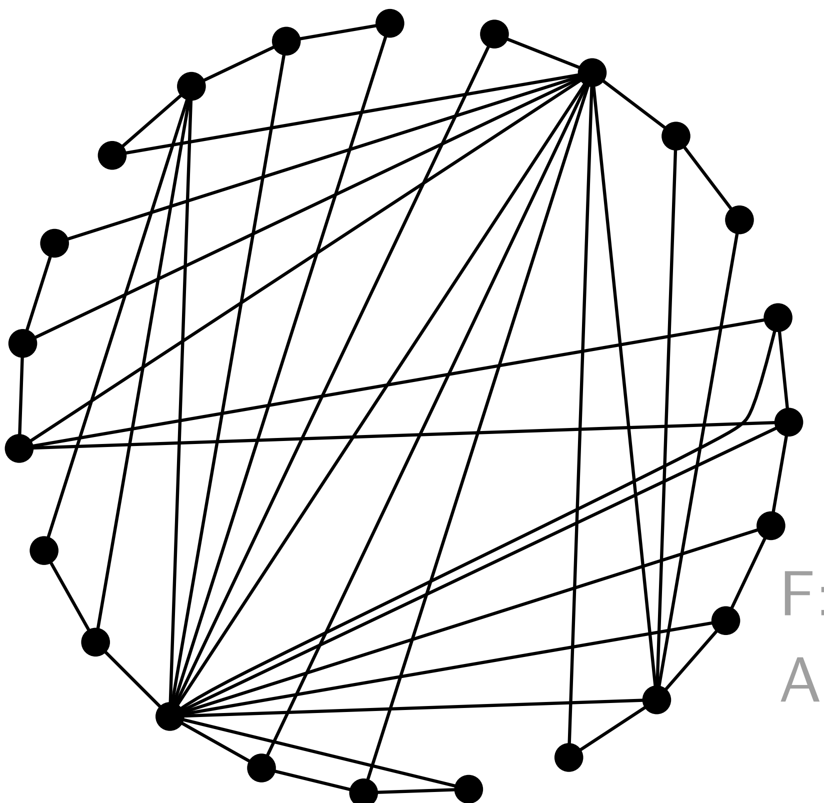
A: [Alam et al. '16]

Minimize crossings of bundles instead of edges!

- gen. layouts: NP-c for fixed [F] and variable [A] embeddings.
- fixed embedding: 10-apx for circular, and $O(1)$-apx for gen. layouts [F]

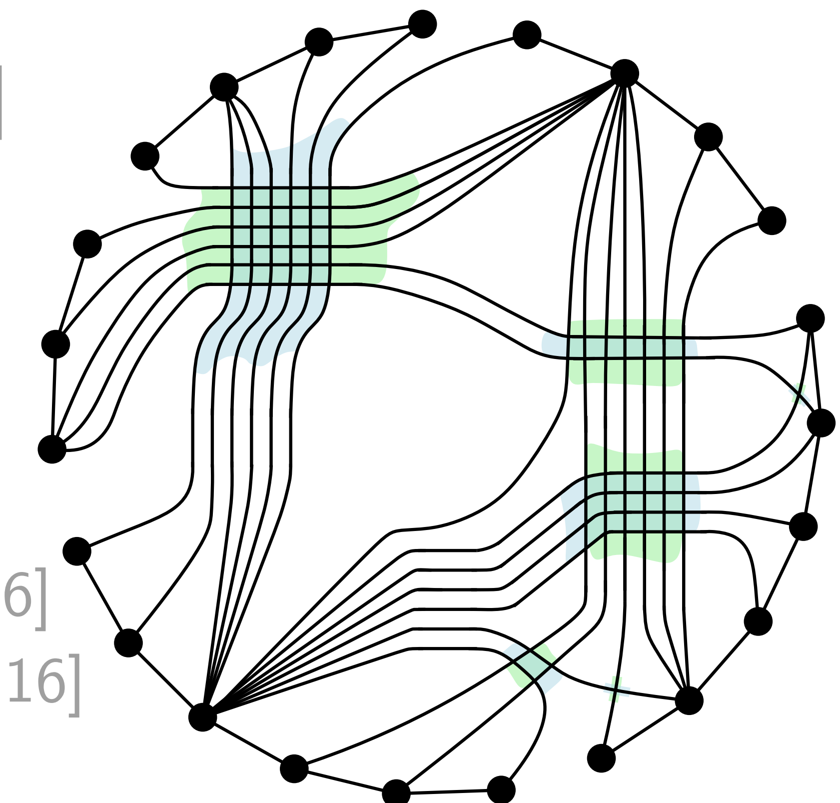
There is an FPT algorithm for deciding whether a graph admits a circular layout with $k$ crossings. [Bannister, Eppstein '14]

# Motivation



[Holten '06]

Bundle the drawing



F: [Fink et al. '16]

A: [Alam et al. '16]

Minimize crossings of bundles instead of edges!

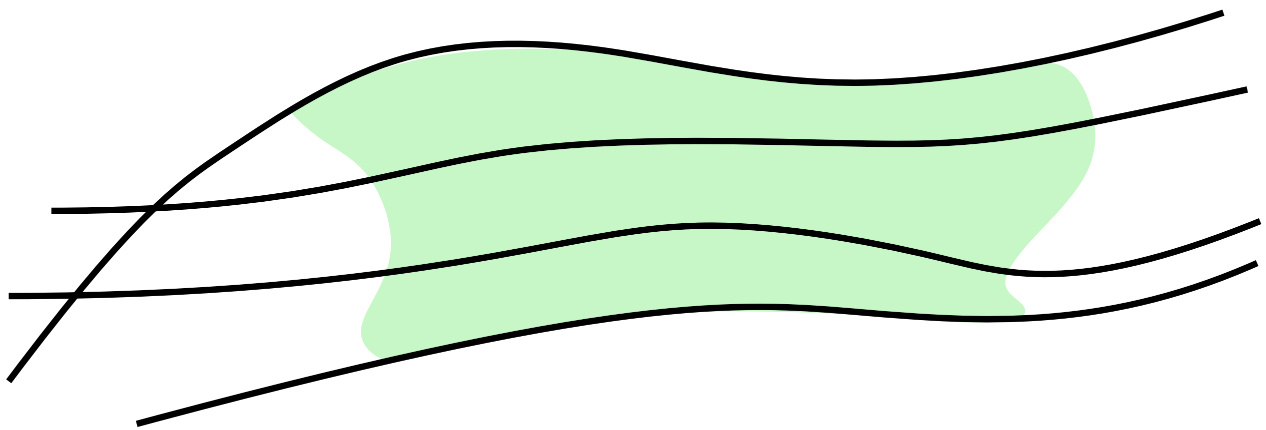
gen. layouts: NP-c for fixed [F] and variable [A] embeddings.

fixed embedding: 10-apx for circular, and $O(1)$-apx for gen. layouts [F]

**Ques.** Is there an FPT algorithm for deciding whether a graph admits a circular layout with $k$ **bundled crossings**? [A]

# Bundled Crossing

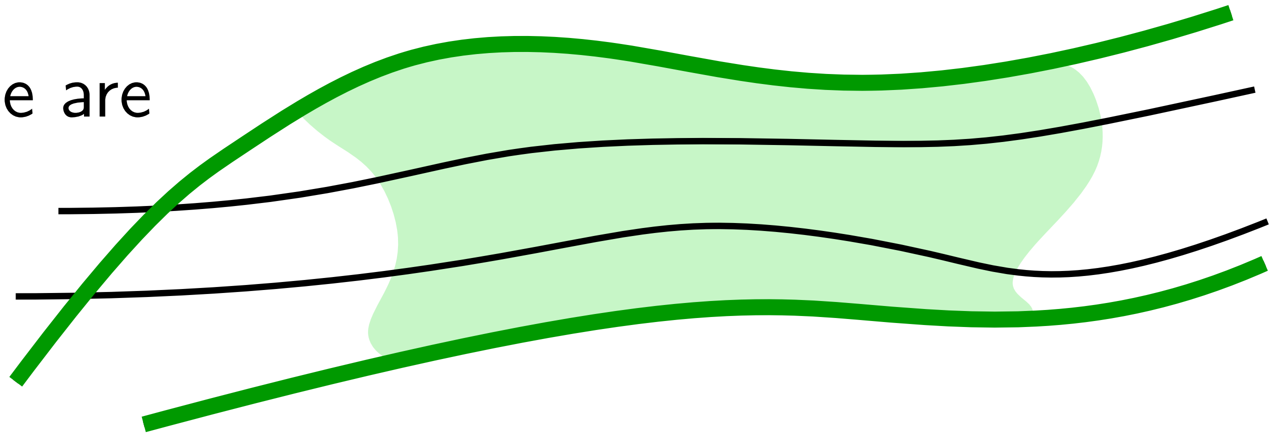
A **bundle** is a set of pieces of edges that travel in parallel in the drawing.



# Bundled Crossing

A **bundle** is a set of pieces of edges that travel in parallel in the drawing.

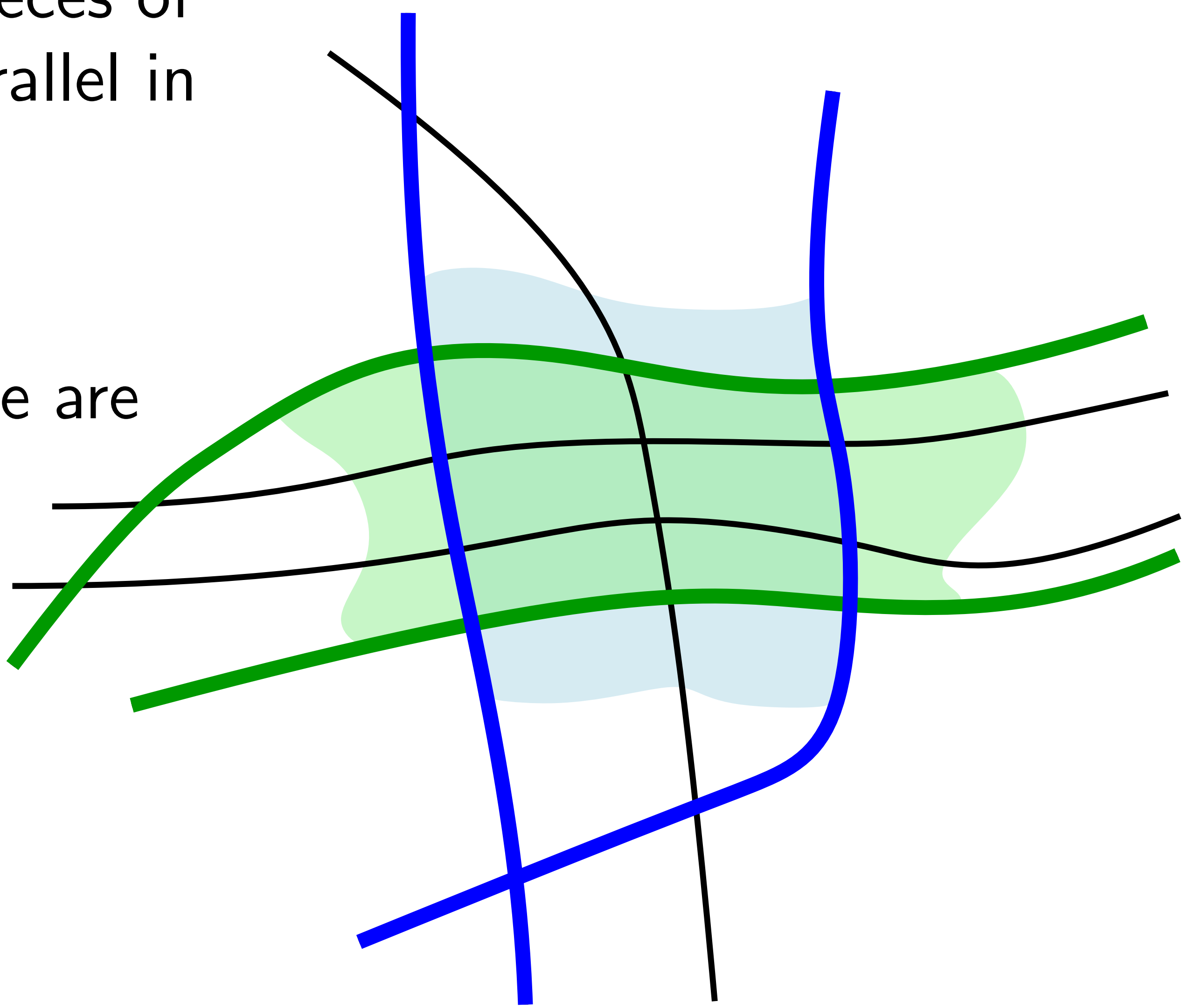
Outer edges of a bundle are called **frame edges**



# Bundled Crossing

A **bundle** is a set of pieces of edges that travel in parallel in the drawing.

Outer edges of a bundle are called **frame edges**

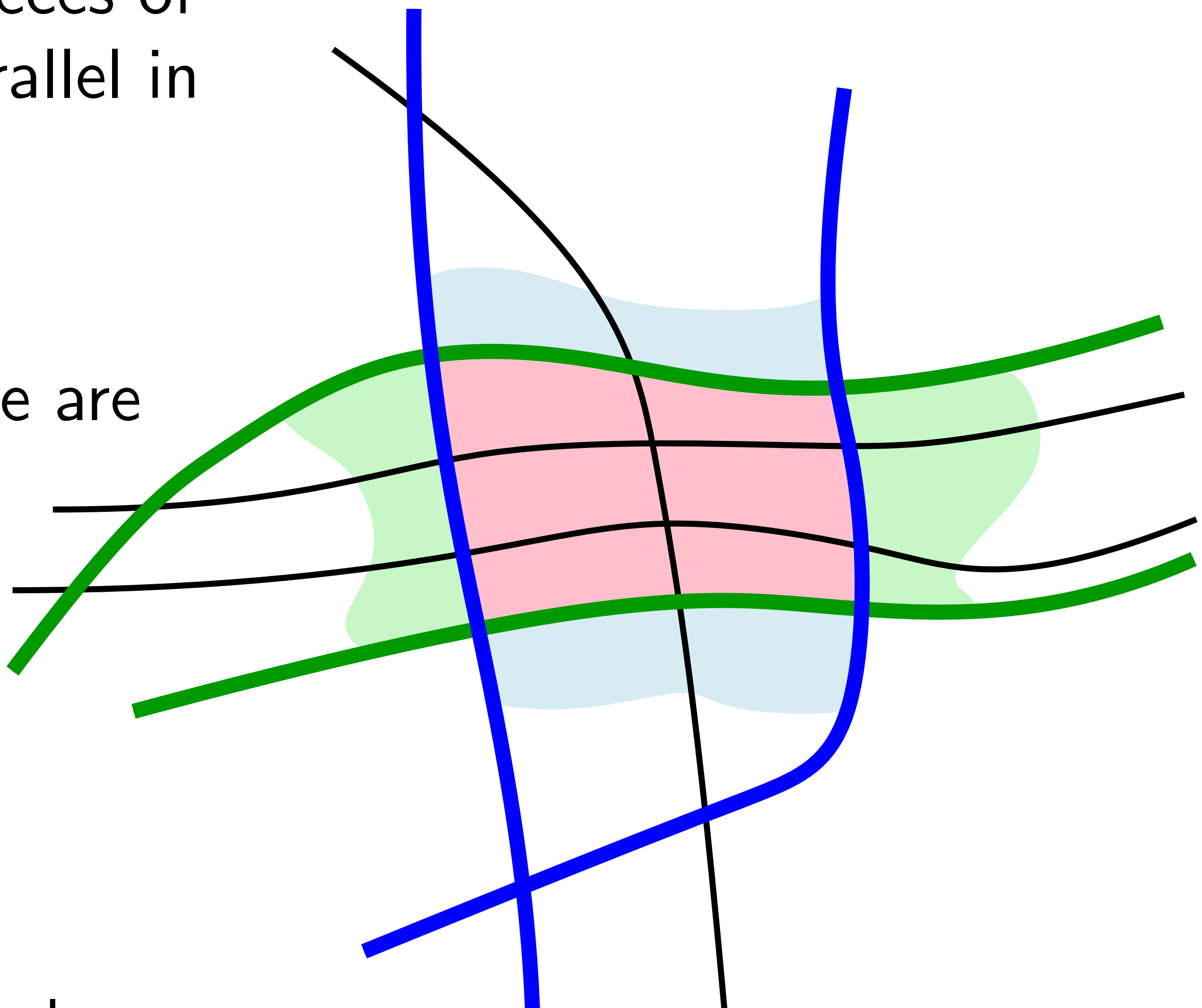


# Bundled Crossing

A **bundle** is a set of pieces of edges that travel in parallel in the drawing.

Outer edges of a bundle are called **frame edges**

A **bundled crossing** is a set of crossings inside the region bounded by the frame edges.



# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^\circ(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^{\circ}(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

**Thm.** Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.

→ resolves an open problem of [Alam et al. 2016]

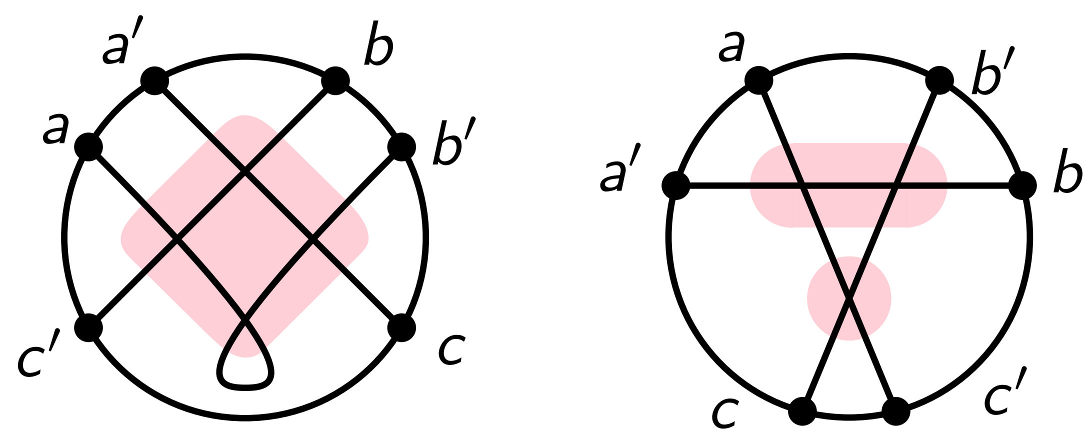
# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^{\circ}(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

**Thm.** Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.

resolves an open problem of [Alam et al. 2016]

Remark on simple vs. non-simple: consider $K_{3,3}$



# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^\circ(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

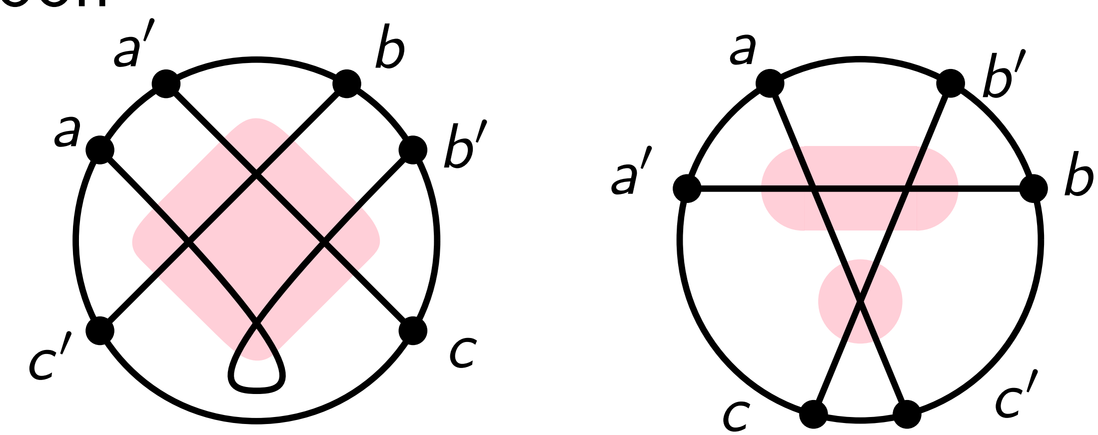
**Thm.** Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.

resolves an open problem of [Alam et al. 2016]

Remark on simple vs. non-simple: consider $K_{3,3}$

Non-simple $\rightsquigarrow$ orientable graph genus [Alam et al. 2016]

... more on this soon



# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^\circ(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

**Thm.** Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.

resolves an open problem of [Alam et al. 2016]

Other results (not covered in this talk, see the paper!):

# Bundled Crossing Minimization

**Def.** For a given graph $G$
the circular bundled crossing number $\mathrm{bc}^{\circ}(G)$ of $G$ is
the minimum number of bundled crossings
over all possible bundlings
of all possible simple circular layouts of $G$.

resolves open problem of [Fink et al.; 2016]

**Thm.** Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.

resolves an open problem of [Alam et al. 2016]

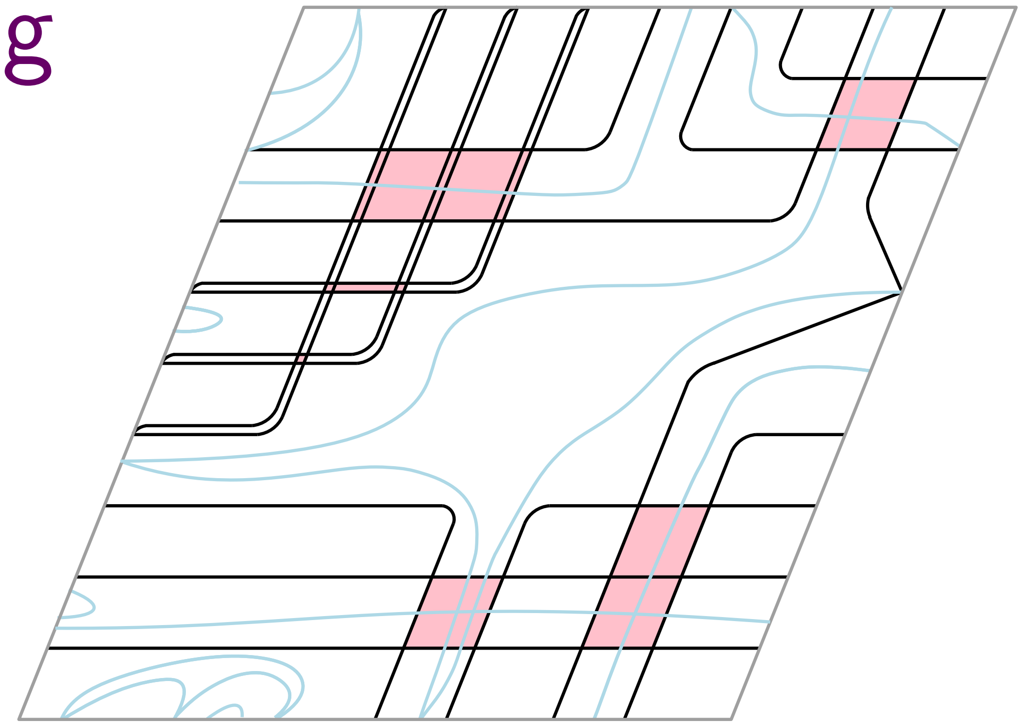
Other results (not covered in this talk, see the paper!):

**Thm.** For general layouts, on inputs $(G, k)$, deciding whether $G$ has a simple drawing with $k$ bundled crossings is NPc. For non-simple, this is FPT in $k$ (via genus).

**Obs.** For circular layouts, on inputs $(G, k)$, deciding whether $G$ has a (non-simple) circular drawing with $k$ bundled crossings is FPT in $k$ (via genus).

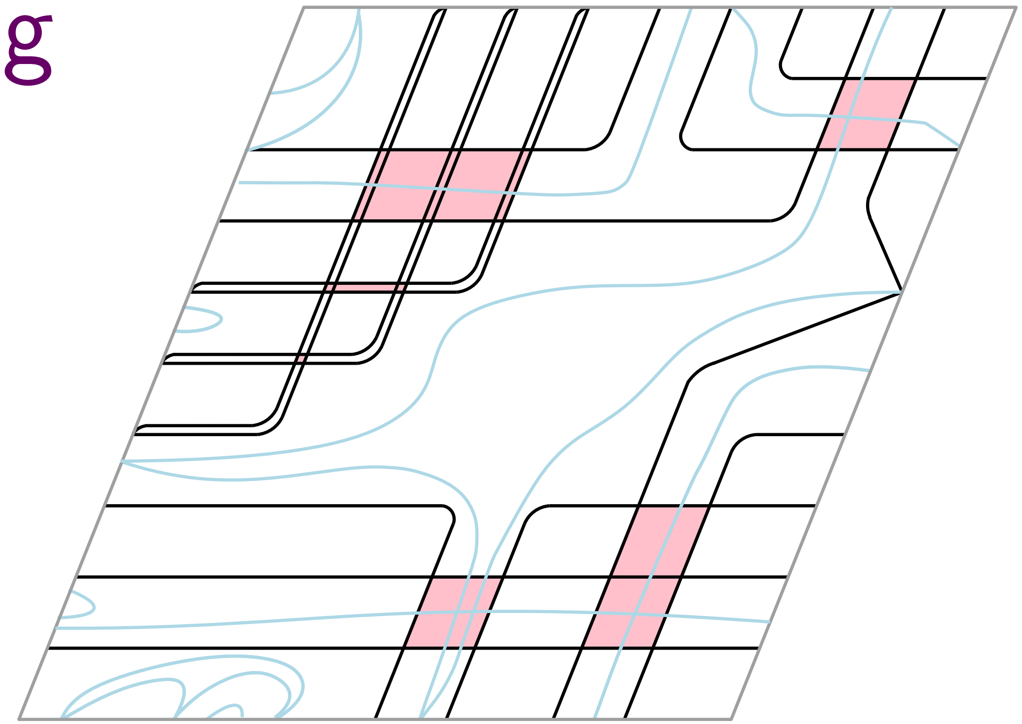
# Structure of a drawing

Consider a drawing with $k$ bundled crossings and observe that:



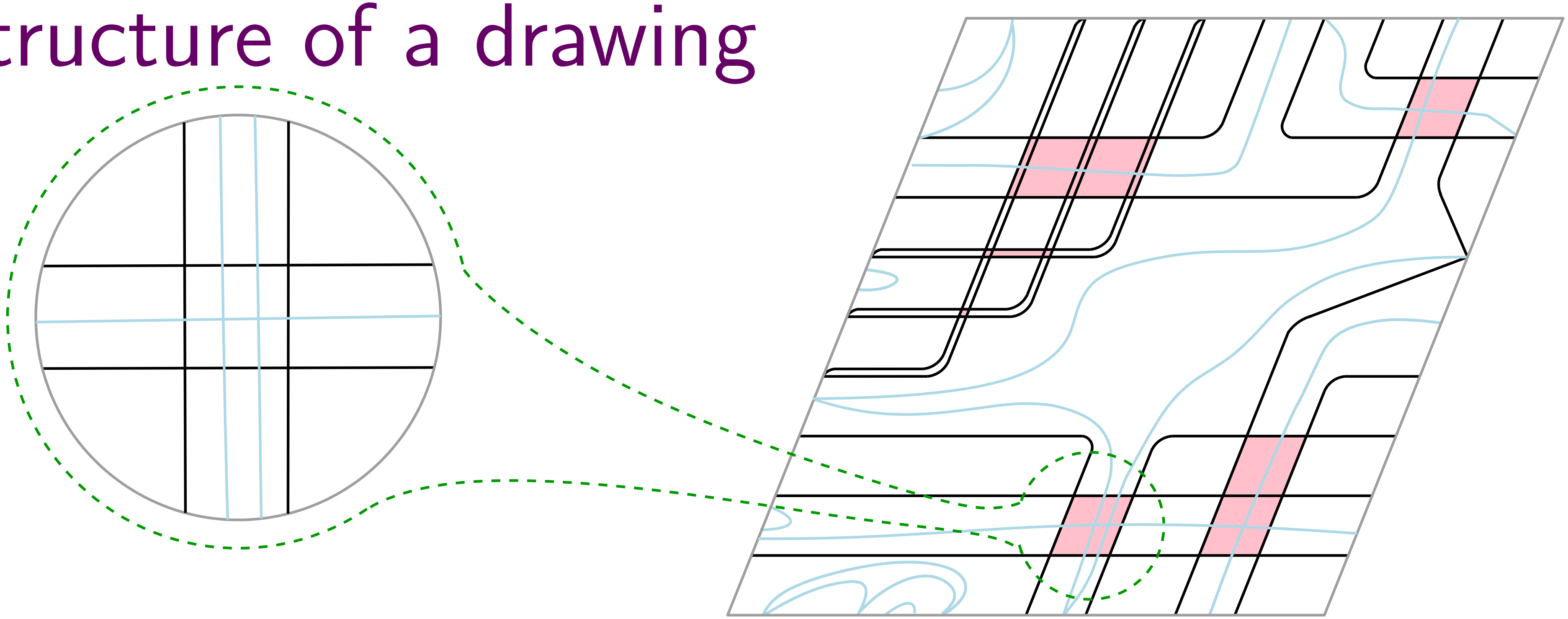
# Structure of a drawing

Consider a drawing with $k$ bundled crossings and observe that:



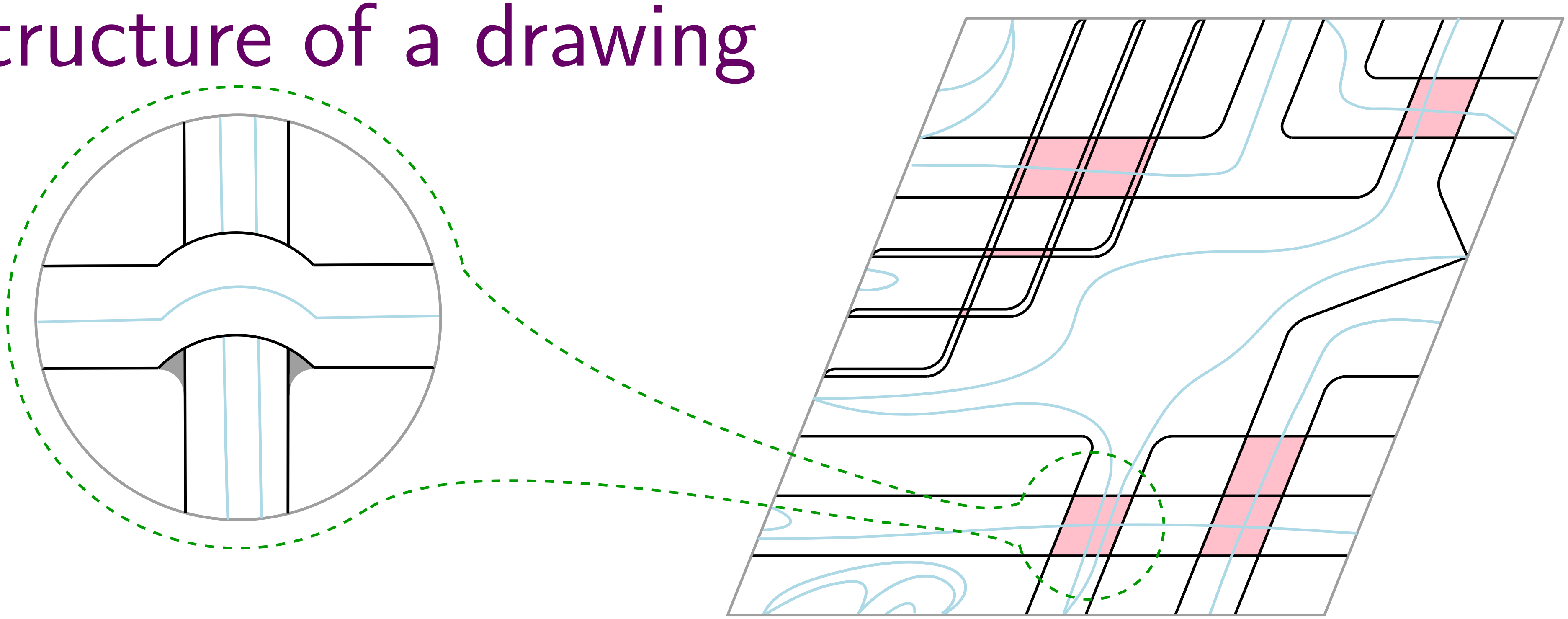
- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.

# Structure of a drawing



- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$

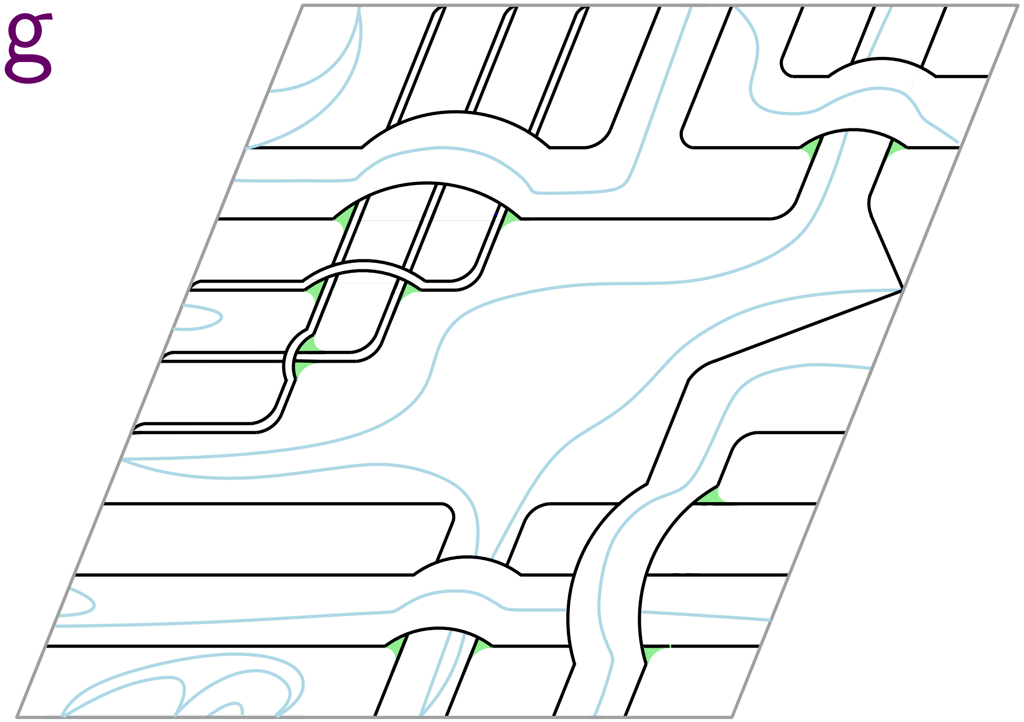
# Structure of a drawing



- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$

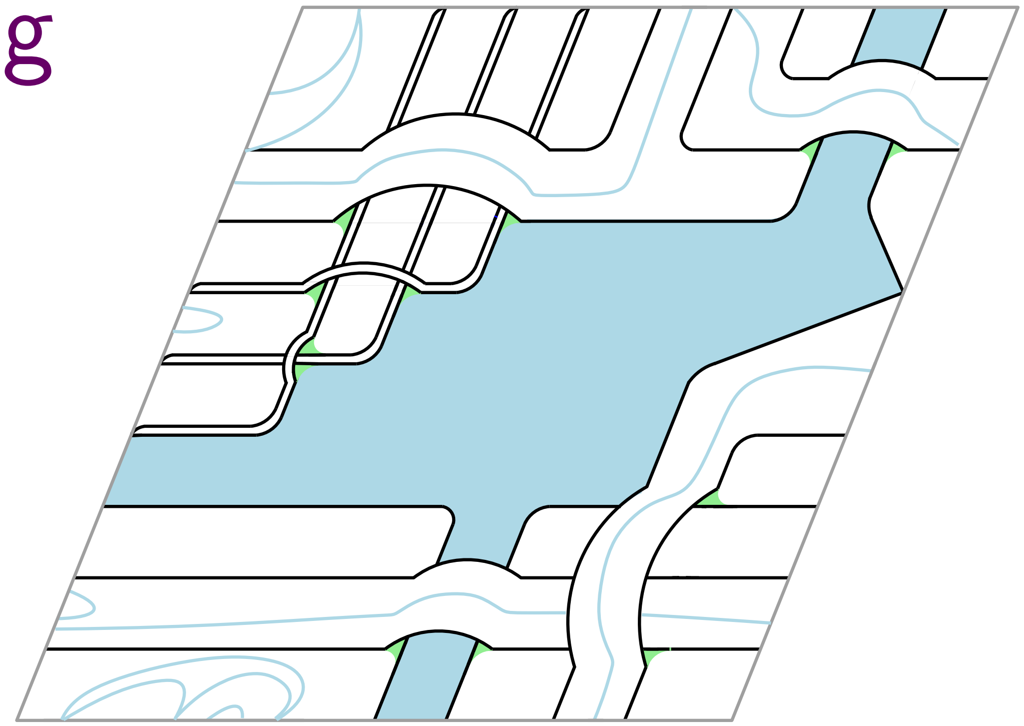
# Structure of a drawing

Consider a drawing with $k$ bundled crossings and observe that:



- At most $k$ bundled crossings $\Longrightarrow$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$

# Structure of a drawing

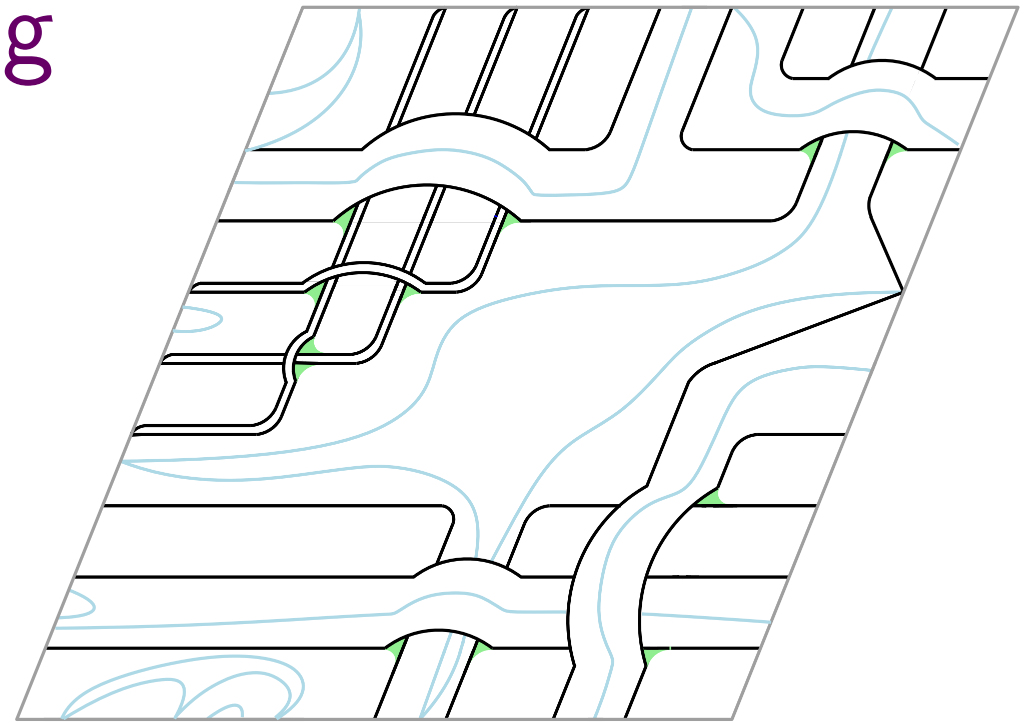


Consider a drawing with $k$ bundled crossings and observe that:

- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$
- and subdivide the surface into **regions**.

# Structure of a drawing

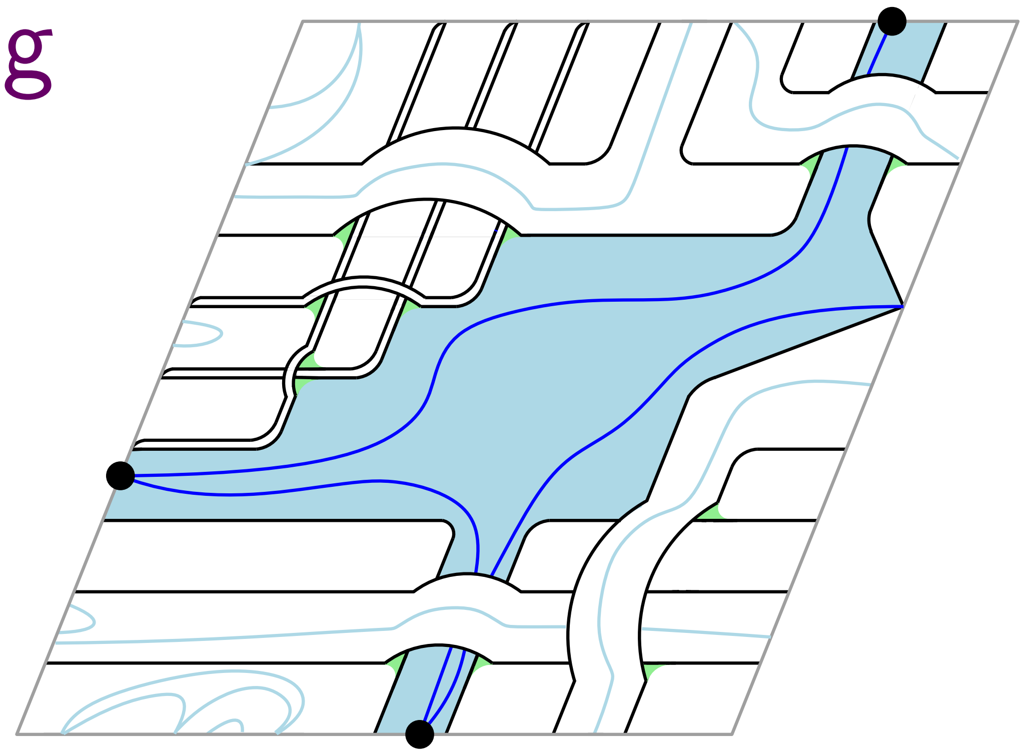
Consider a drawing with $k$ bundled crossings and observe that:



- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus $k$**
- and subdivide the surface into **regions**.
- Other edges/vertices of the graph partitioned into these regions.

# Structure of a drawing

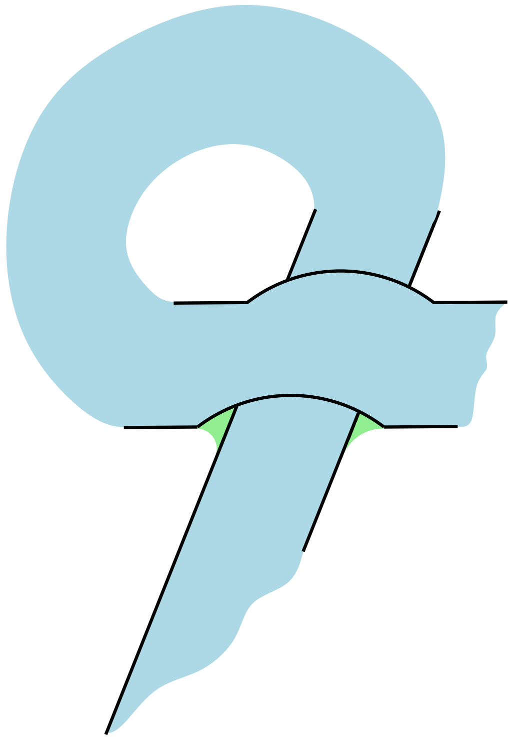
Consider a drawing
with $k$ bundled crossings
and observe that:



- At most $k$ bundled crossings $\implies$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$
- and subdivide the surface into **regions**.
- Other edges/vertices of the graph partitioned into these regions.
- The graph induced by edges inside a single region has a special outerplanar drawing.

# Structure of a drawing

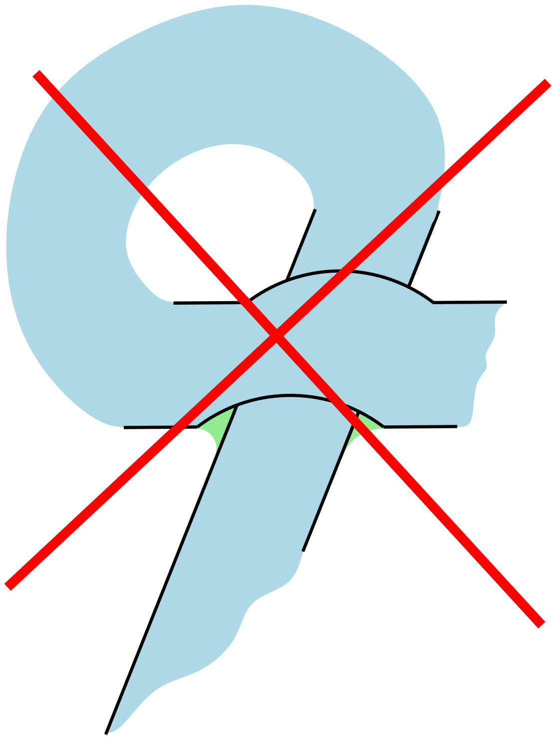
What if a region has
a bridge and a tunnel
corresponding to
the same bundled crossing?



- At most $k$ bundled crossings $\Longrightarrow$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$
- and subdivide the surface into **regions**.
- Other edges/vertices of the graph partitioned into these regions.
- The graph induced by edges inside a single region has a special outerplanar drawing.

# Structure of a drawing

What if a region has
a bridge and a tunnel
corresponding to
the same bundled crossing?

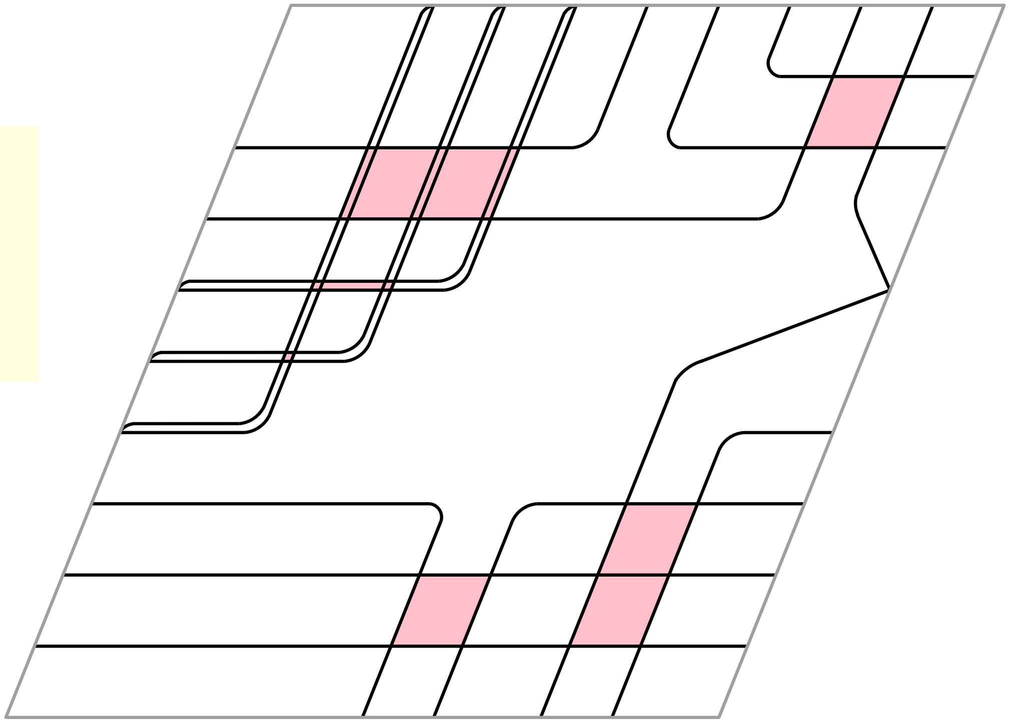


**Lem.** Each region is a topological disk.

- At most $k$ bundled crossings $\Longrightarrow$ at most $4k$ frame edges.
- We can "lift" the drawing onto a **surface of genus** $k$
- and subdivide the surface into **regions**.
- Other edges/vertices of the graph partitioned into these regions.
- The graph induced by edges inside a single region has a special outerplanar drawing.

# The Algorithm

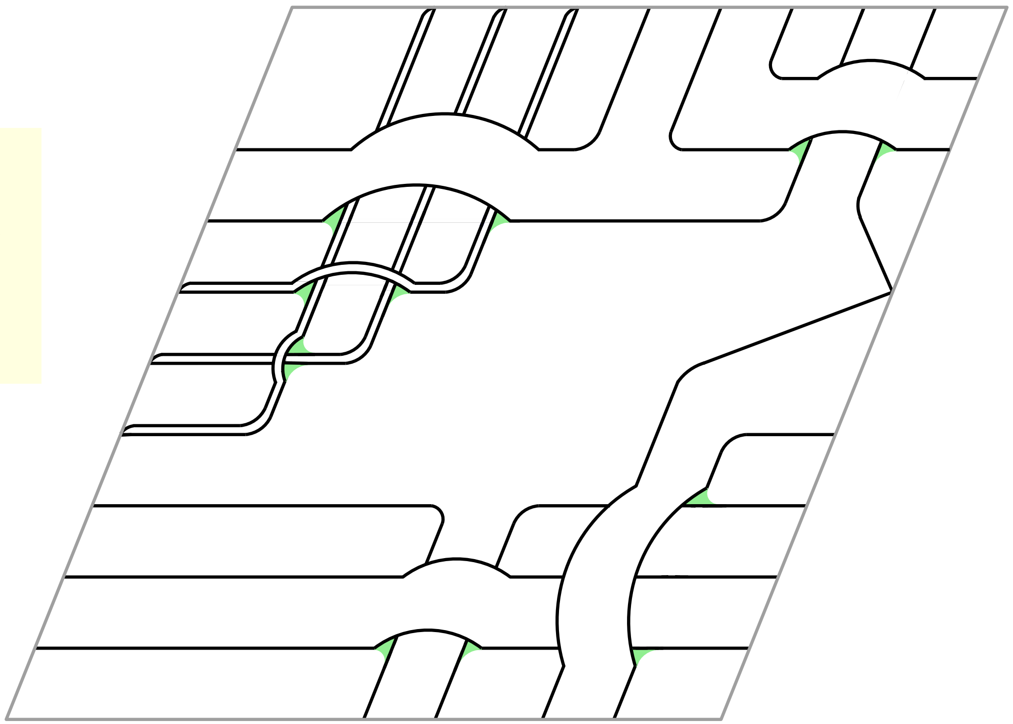
**Thm.**
Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.



- Guess the drawing of at most $4k$ frame edges and their bundling.

# The Algorithm

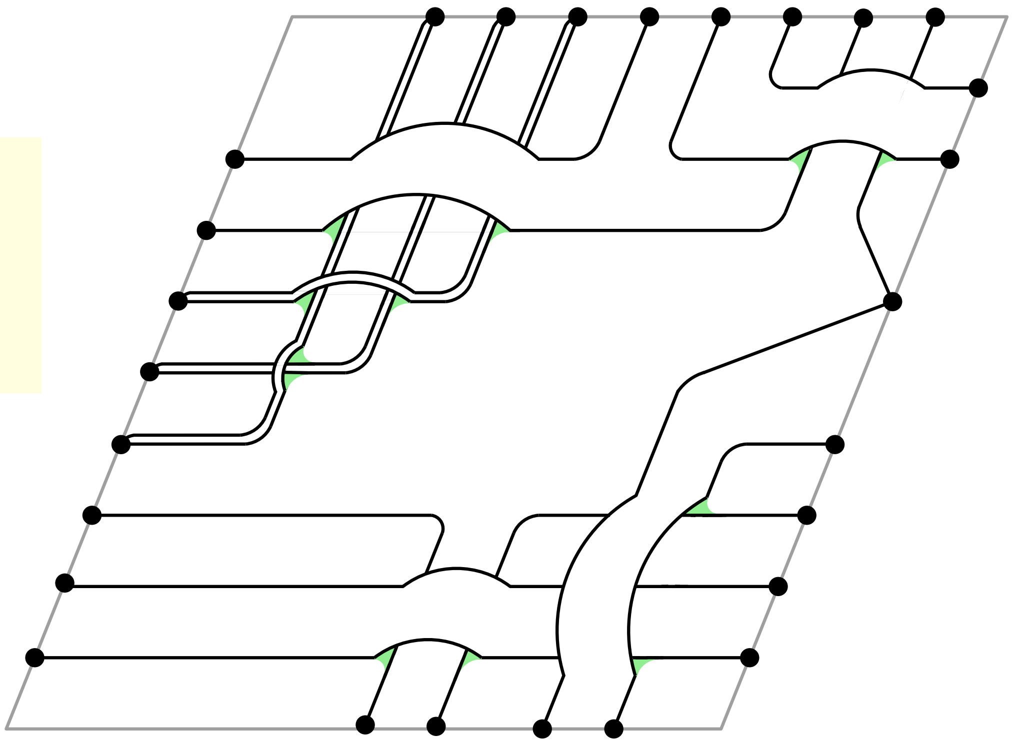
**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.



- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.

# The Algorithm

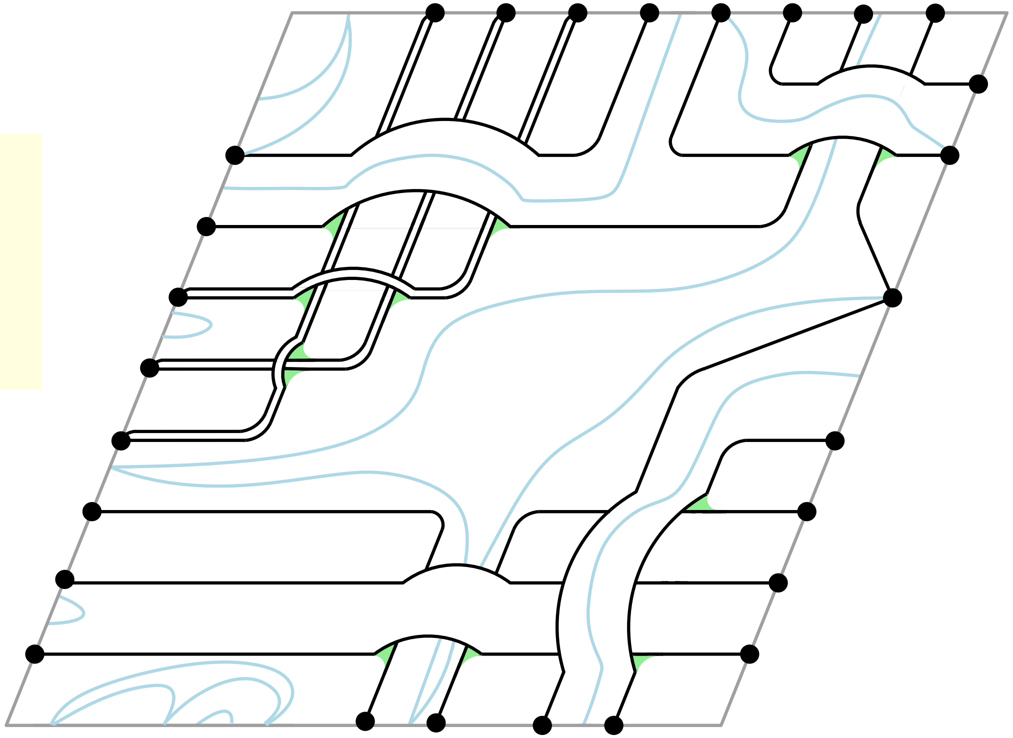
**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.



- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.

# The Algorithm

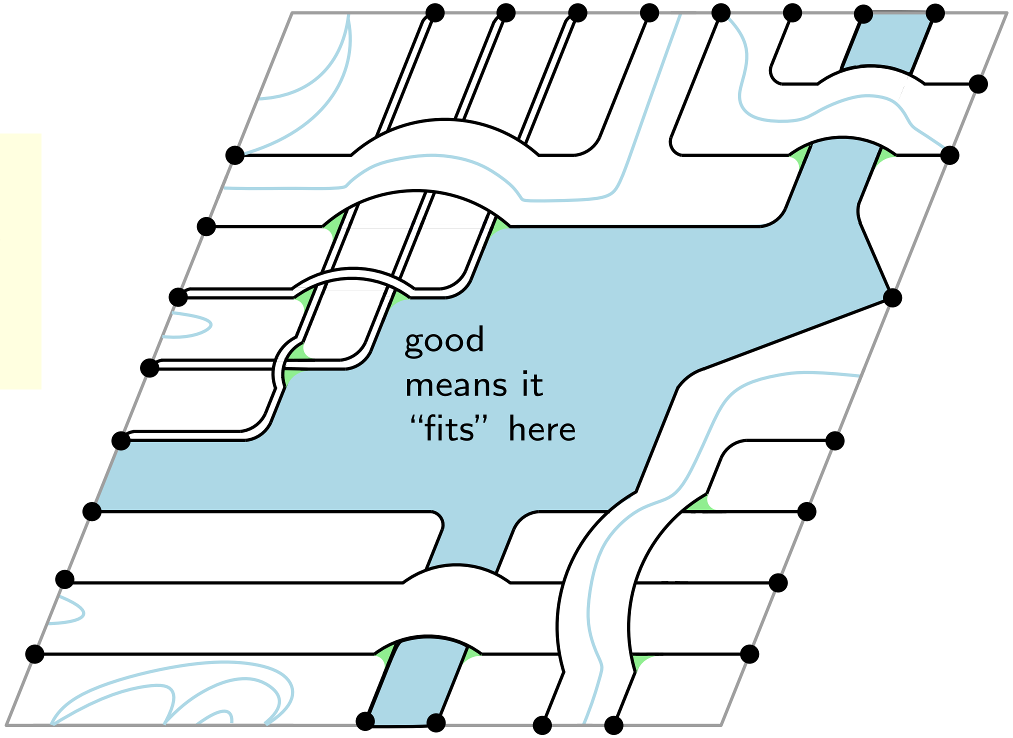
**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.



- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
    - Map the edges of the graph to the guessed frame edges.
    - Partition the edges and vertices into the regions.

# The Algorithm

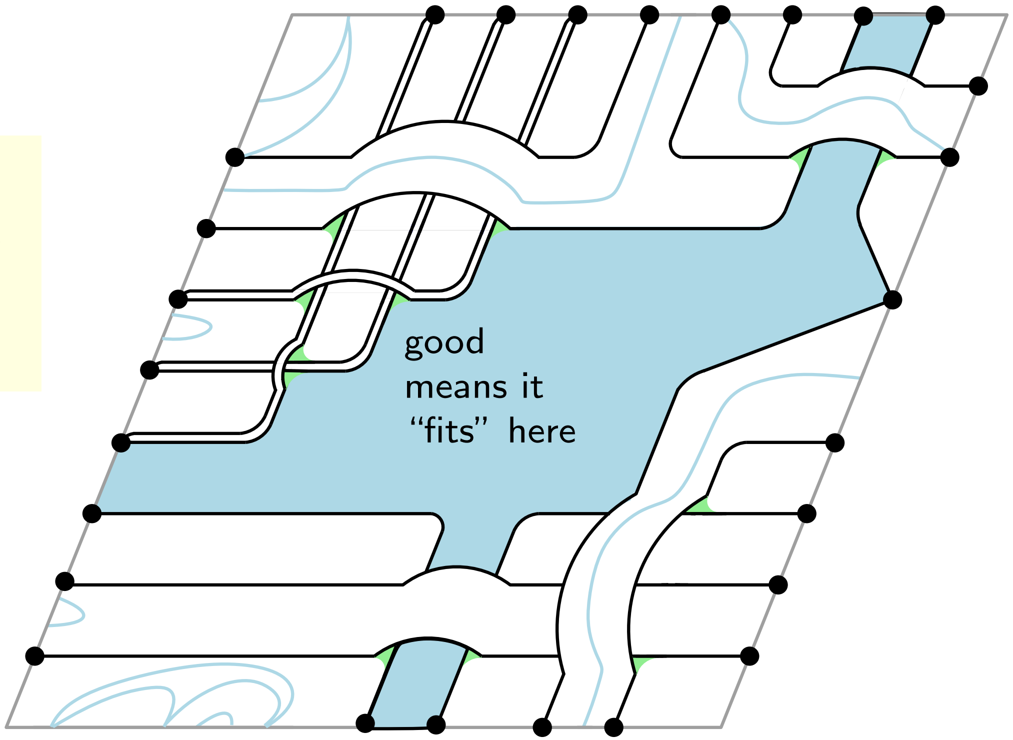
**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.




- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

# The Algorithm

**Thm.**
Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.



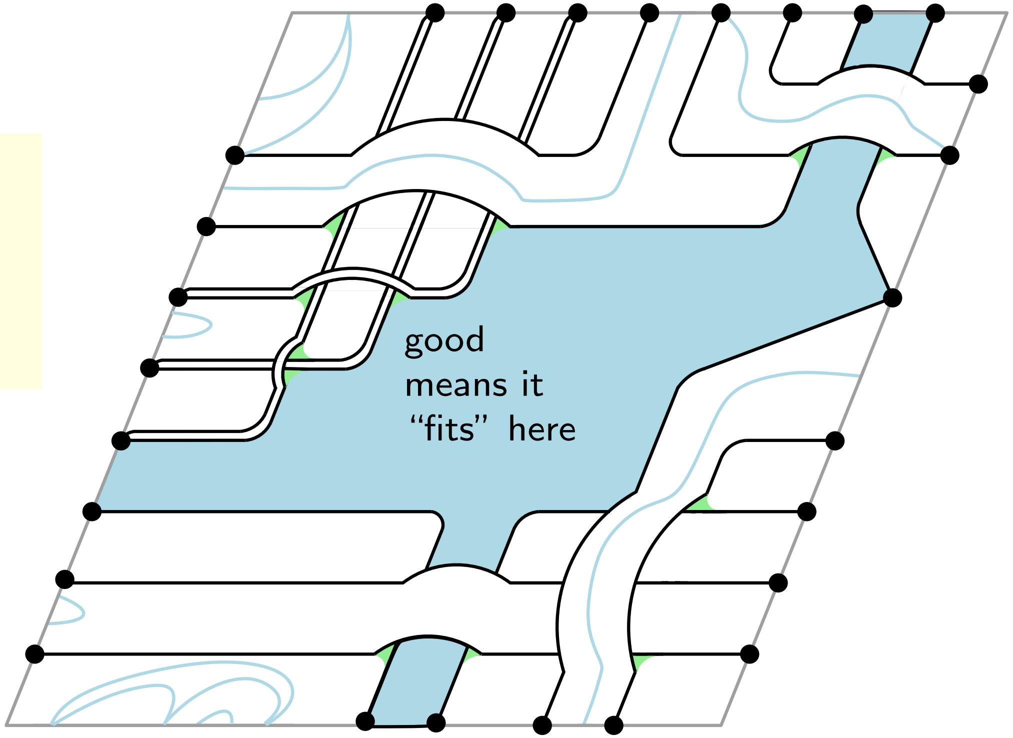

- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

$\mathrm{MSO}_2$

# The Algorithm

**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.

But what is $\mathrm{MSO}_2$ again?

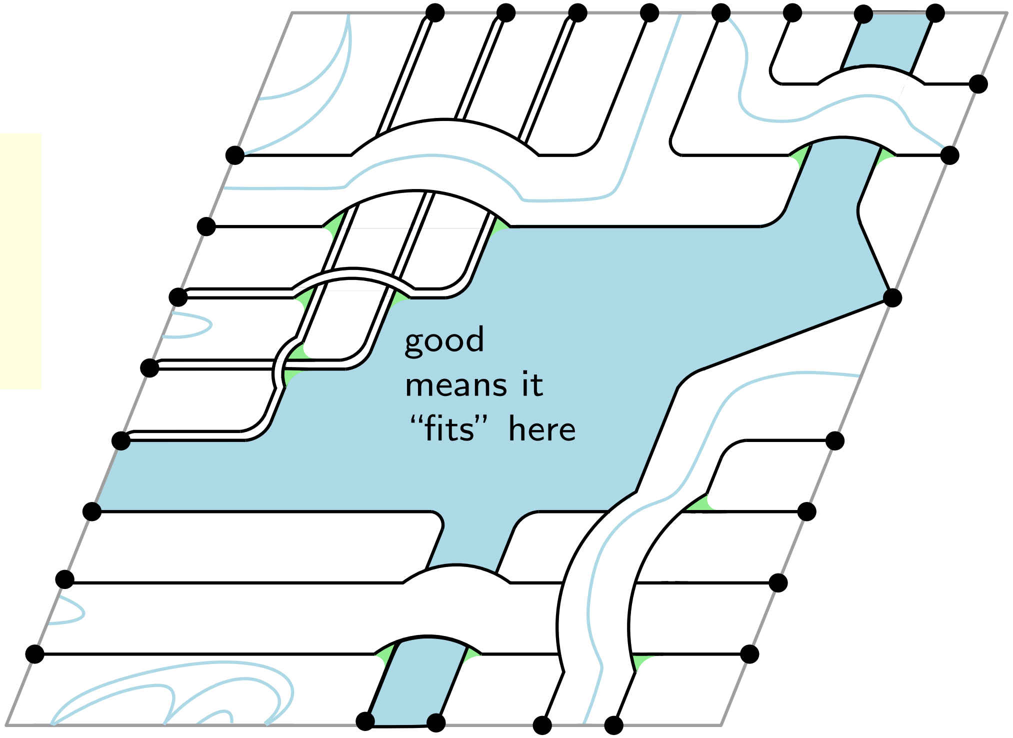



- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

$\mathrm{MSO}_2$

# The Algorithm

**Thm.**
Deciding whether $\mathrm{bc}^{\circ}(G) = k$ is FPT in $k$.




But what is $\mathrm{MSO}_2$ again?

- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

$\mathrm{MSO}_2$

# Monadic Second Order Logic ($\mathrm{MSO}_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) =$$
$$(\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

# Monadic Second Order Logic ($MSO_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) =$$
$$(\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

**Thm (Courcelle):** If a property $P$ is expressed as $\varphi \in \mathrm{MSO}_2$, then for every graph $G$ with treewidth at most $t$, $P$ can be tested in time $O(f(t, |\varphi|)(n+m))$ for a computable function $f$.

# Monadic Second Order Logic ($\mathrm{MSO}_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) = (\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

**Thm (Courcelle):** If a property $P$ is expressed as $\varphi \in \mathrm{MSO}_2$, then for every graph $G$ with treewidth at most $t$, $P$ can be tested in time $O(f(t, |\varphi|)(n+m))$ for a computable function $f$.

# Monadic Second Order Logic ($\mathrm{MSO}_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) =$$

$$(\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

**Thm (Courcelle):** If a property $P$ is expressed as $\varphi \in \mathrm{MSO}_2$, then for every graph $G$ with treewidth at most $t$, $P$ can be tested in time $O(f(t, |\varphi|)(n + m))$ for a computable function $f$.

Since our regions induce outerplanar graphs, we have treewidth at most $8k + 2$ where $k$ is the number of bundled crossings.

# Monadic Second Order Logic ($\mathrm{MSO}_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) =$$
$$(\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

**Thm (Courcelle):** If a property $P$ is expressed as $\varphi \in \mathrm{MSO}_2$, then for every graph $G$ with treewidth at most $t$, $P$ can be tested in time $O(f(t, |\varphi|)(n+m))$ for a computable function $f$.

But, what about?

- Test graphs in each region for a *good* outerplanar drawing.

# Monadic Second Order Logic ($MSO_2$)

$$\textsc{Partition}(E; E_0, \ldots, E_\gamma) = (\forall e \in E)\Big[\Big(\bigvee_{i=0}^{\gamma} e \in E_i\Big) \wedge \Big(\bigwedge_{i \neq j} \neg(e \in E_i \wedge e \in E_j)\Big)\Big].$$

**Thm (Courcelle):** If a property $P$ is expressed as $\varphi \in MSO_2$, then for every graph $G$ with treewidth at most $t$, $P$ can be tested in time $O(f(t, |\varphi|)(n+m))$ for a computable function $f$.

But, what about?

- Test graphs in each region for a *good* outerplanar drawing.

This can be stated in $MSO_2$ via a mechanism of *MSO-definition schemes*, and the *Backwards Translation Theorem* [Courcelle, Engelfriet; 2012]

# Monadic Second Order Logic ($MSO_2$)

**Theorem 7.10 (Backwards Translation Th**[illegible] Let $\mathcal{D}$ be a $k$-copying $C_r MS$-definition scheme of type $\mathcal{R} \to \mathcal{R}'$ [illegible] of parameters $\mathcal{W}$. Let $\mathcal{X}$ be a finite set of set variables and $\mathcal{Y} = \{y$ [illegible] be a set of first-order variables. For every $\beta \in C_r MS(\mathcal{R}', \mathcal{X} \cup \mathcal{Y})$ and [illegible] one can construct a formula $\beta_{\mathbf{i}}^{\mathcal{D}} \in C_r MS(\mathcal{R}, \mathcal{W} \cup \mathcal{X}^{(k)} \cup \mathcal{Y})$ such tha[illegible] $S \in STR^{c}(\mathcal{R})$, every $\mathcal{W}$-assignment $\gamma$, every $\mathcal{X}^{(k)}$-assignment $\eta$, and e[illegible]signment $\mu$, all of them in $S$, we have:

$(S, \gamma \cup \eta \cup \mu) \models \beta_{\mathbf{i}}^{\mathcal{D}}$ if a[illegible]

$\widehat{\mathcal{D}}(S, \gamma)$ is de[illegible]

$\eta^{[k]} \cup \mu_{\mathbf{i}}$ [illegible] $\mathcal{Y})$-assignment in $\widehat{\mathcal{D}}(S, \gamma)$, and

$(\widehat{\mathcal{D}}(S$ [illegible] $\mu_{\mathbf{i}}) \models \beta$.

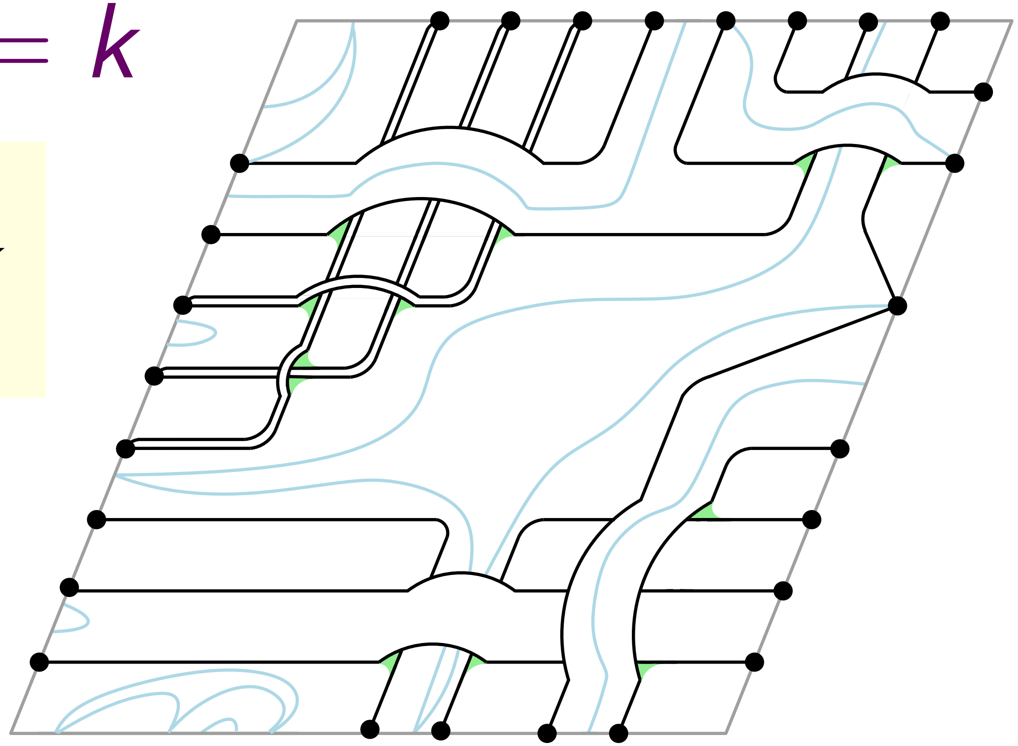
The quantifier-height of $\beta_{\mathbf{i}}^{\mathcal{D}}$ is at most $k \cdot qh(\beta) + qh(\mathcal{D}) + 1$. □

It's a bit technical

This can be stated in $MSO_2$ via a mechanism of *MSO-definition schemes*, and the *Backwards Translation Theorem* [Courcelle, Engelfriet; 2012]

# Testing whether $\mathrm{bc}^\circ = k$

**Thm.**
Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.

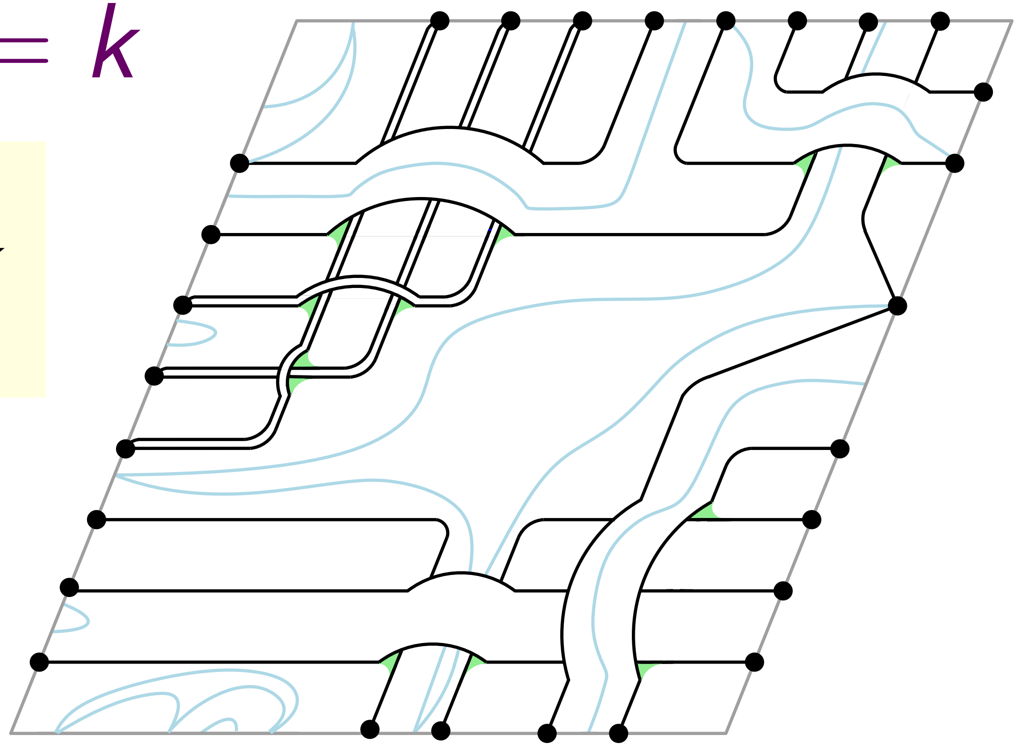


Runtime:

- Guess the drawing of at most $4k$ frame edges and their bundling.
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

# Testing whether $\mathrm{bc}^\circ = k$

**Thm.**
Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.

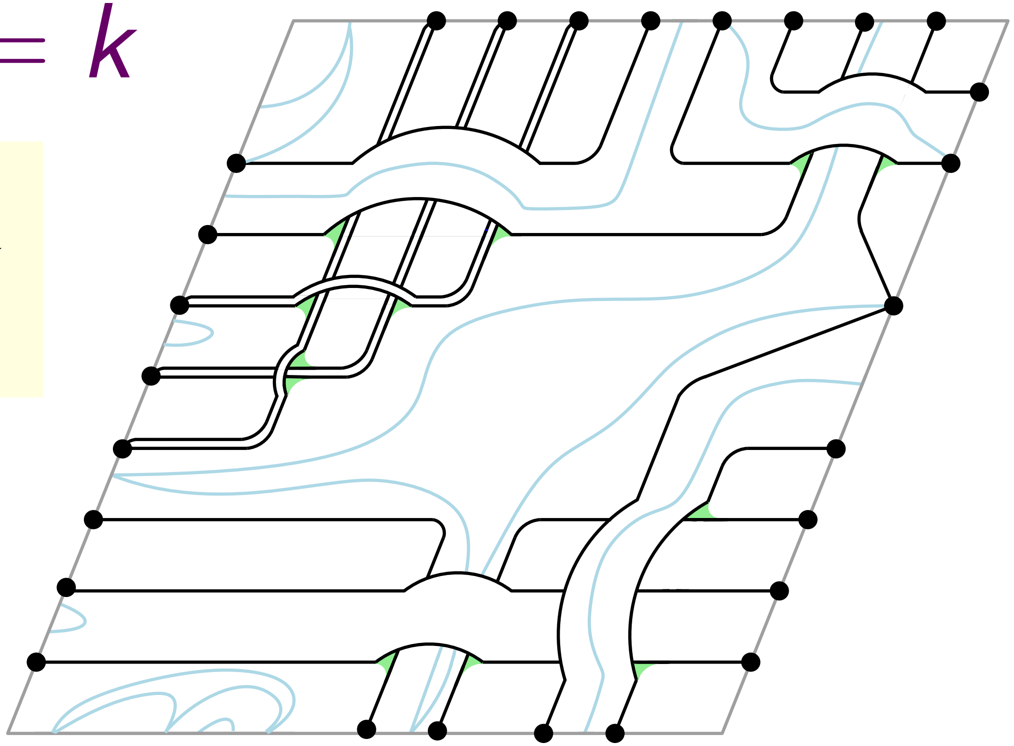


Runtime:

$2^{O(k^2)}$

- Guess the drawing of at most $4k$ frame edges and their bundling. $2^{O(k^2)}$
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

# Testing whether $\mathrm{bc}^\circ = k$



**Thm.**
Deciding whether $\mathrm{bc}^\circ(G) = k$ is FPT in $k$.

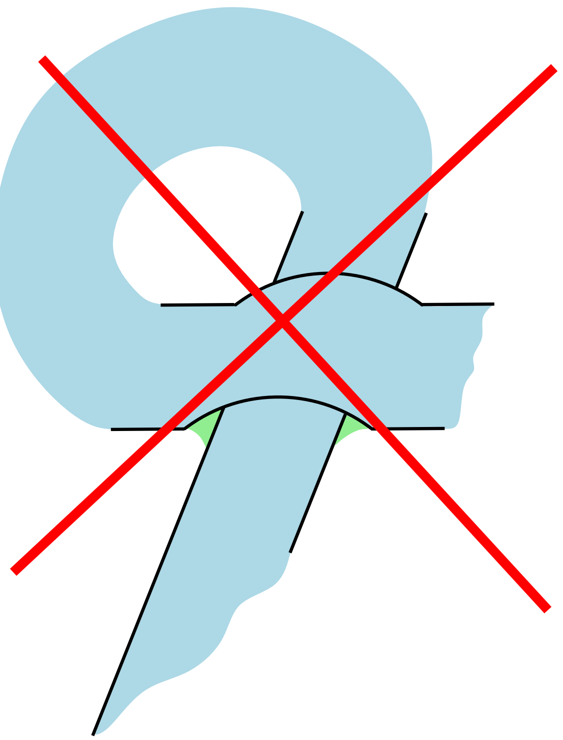
Runtime:

$2^{O(k^2)} f(k)(|V| + |E|)$

- Guess the drawing of at most $4k$ frame edges and their bundling. $2^{O(k^2)}$
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions.
  - Test graphs in each region for a *good* outerplanar drawing.

$\mathrm{MSO}_2$

# Testing whether bc$^\circ = k$



Recall that for correctness of the algorithm we need to show that

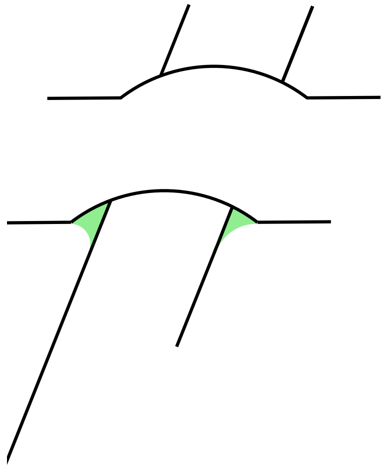
**Thm.** Each region is a topological disk.

- Guess the drawing of at most $4k$ frame edges and their bundling. $2^{O(k^2)}$
- Construct a **surface of genus** $k$ and a subdivision into **regions**.
  - Map the edges of the graph to the guessed frame edges.
  - Partition the edges and vertices into the regions. $\mathrm{MSO}_2$
  - Test graphs in each region for a *good* outerplanar drawing.

# A region is a topological disk

**Lem.** Each region is a topological disk.

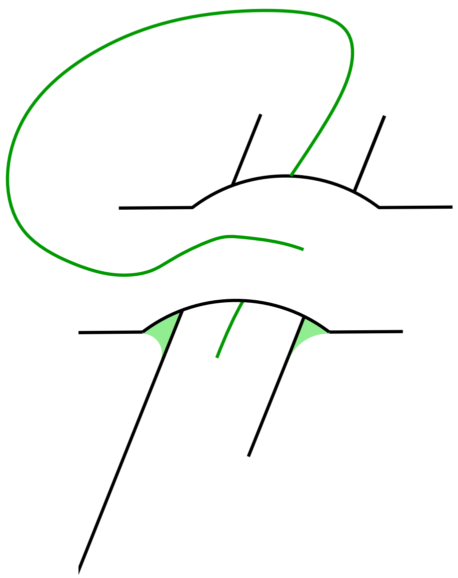
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

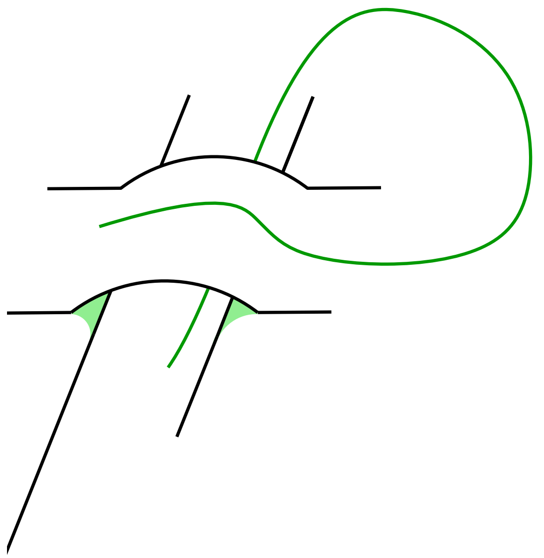
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

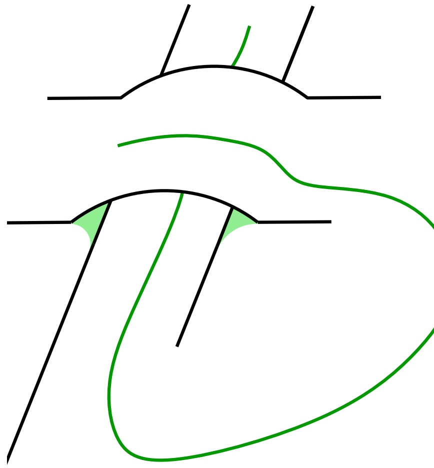
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

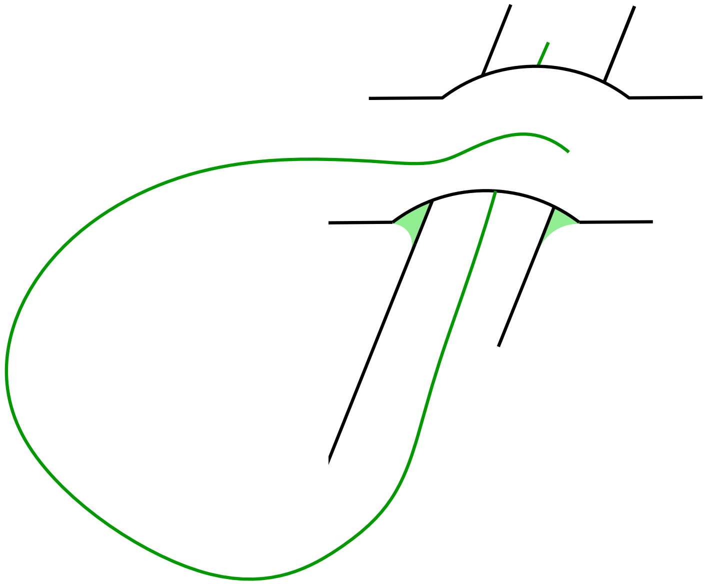
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

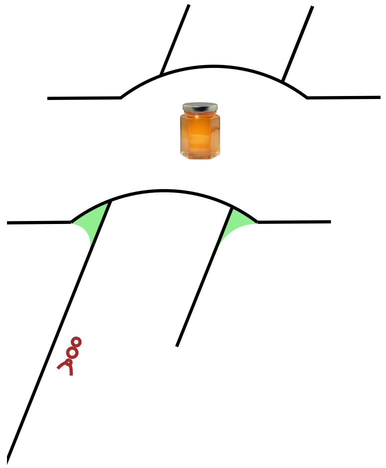
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

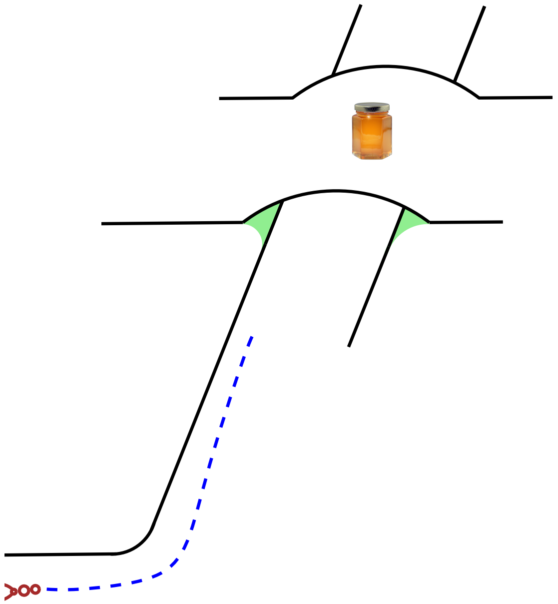


Stick to the right!

# A region is a topological disk

**Lem.** Each region is a topological disk.

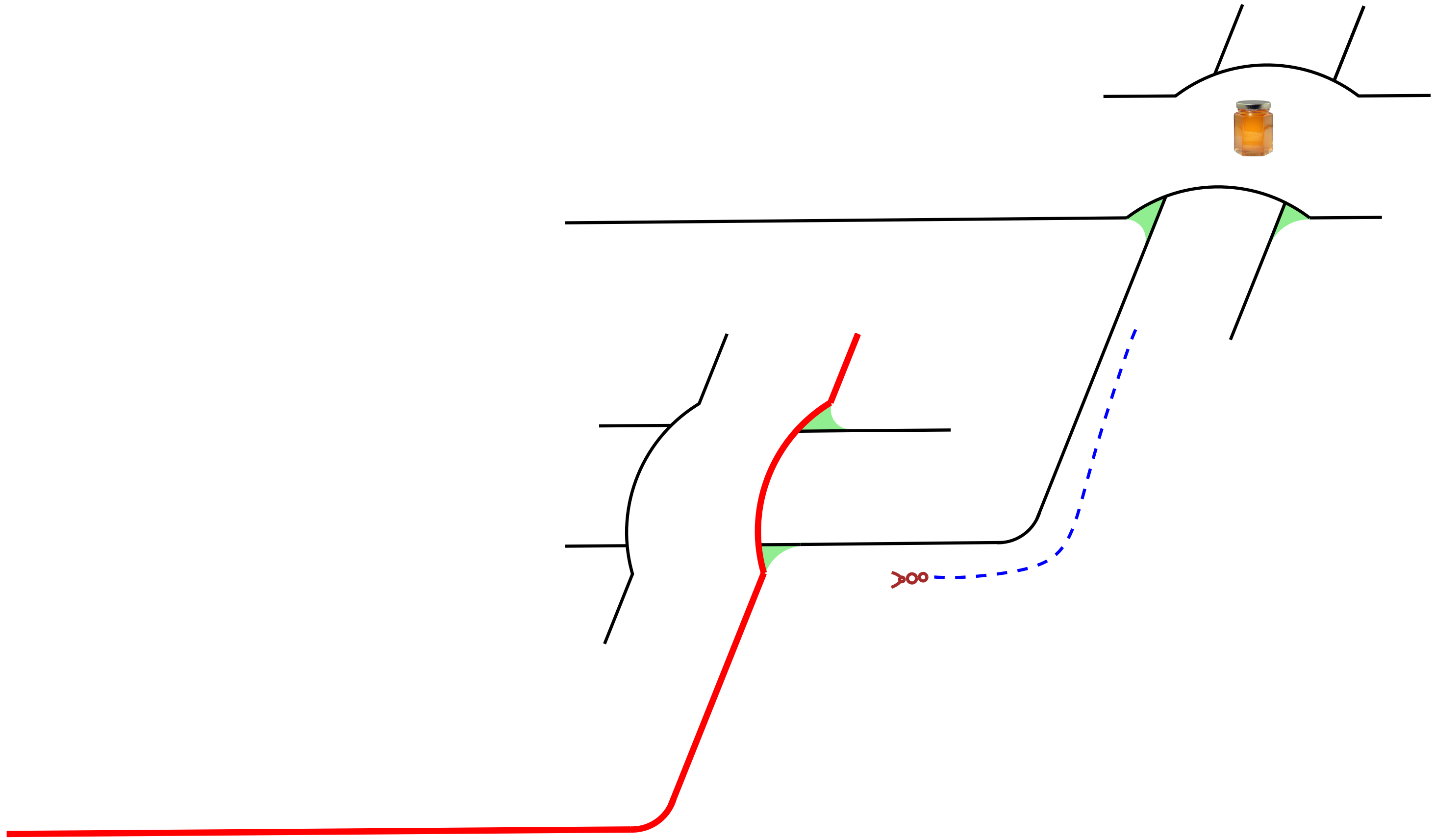
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

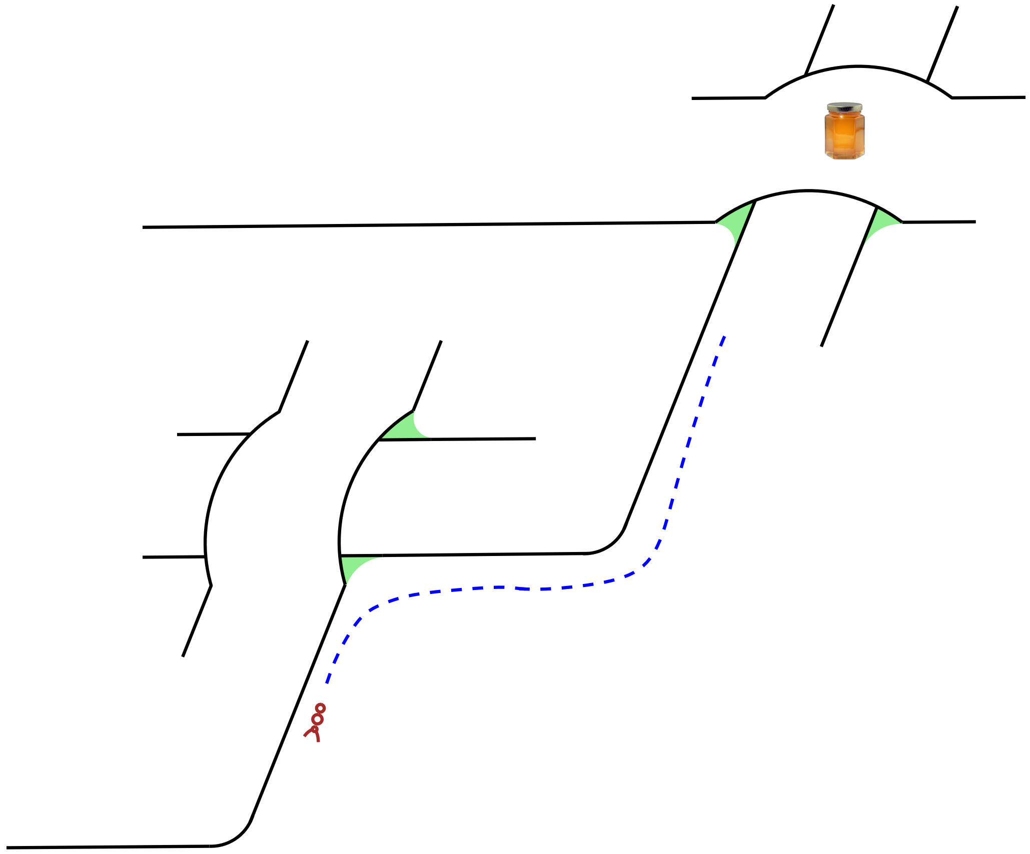
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

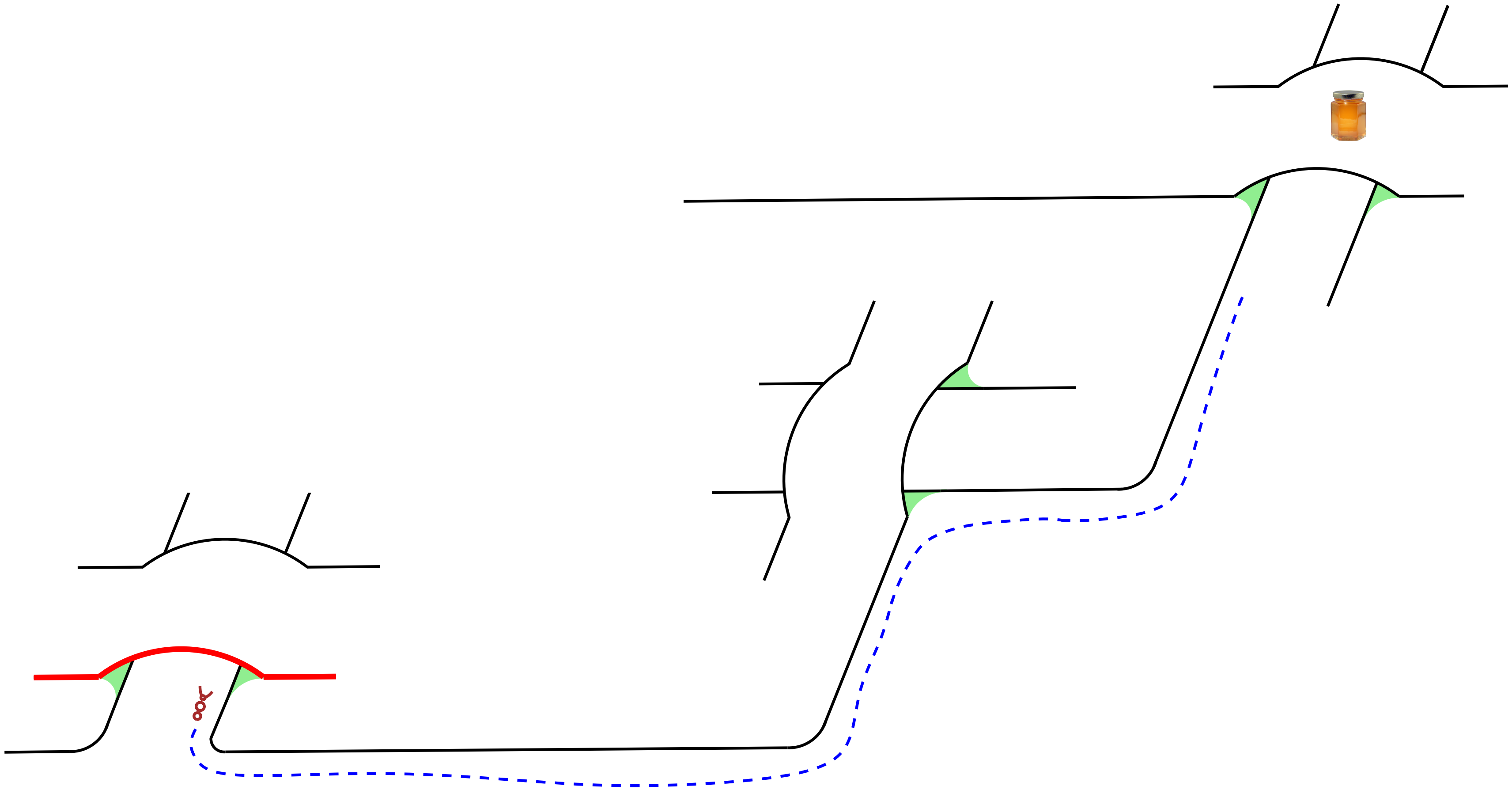
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

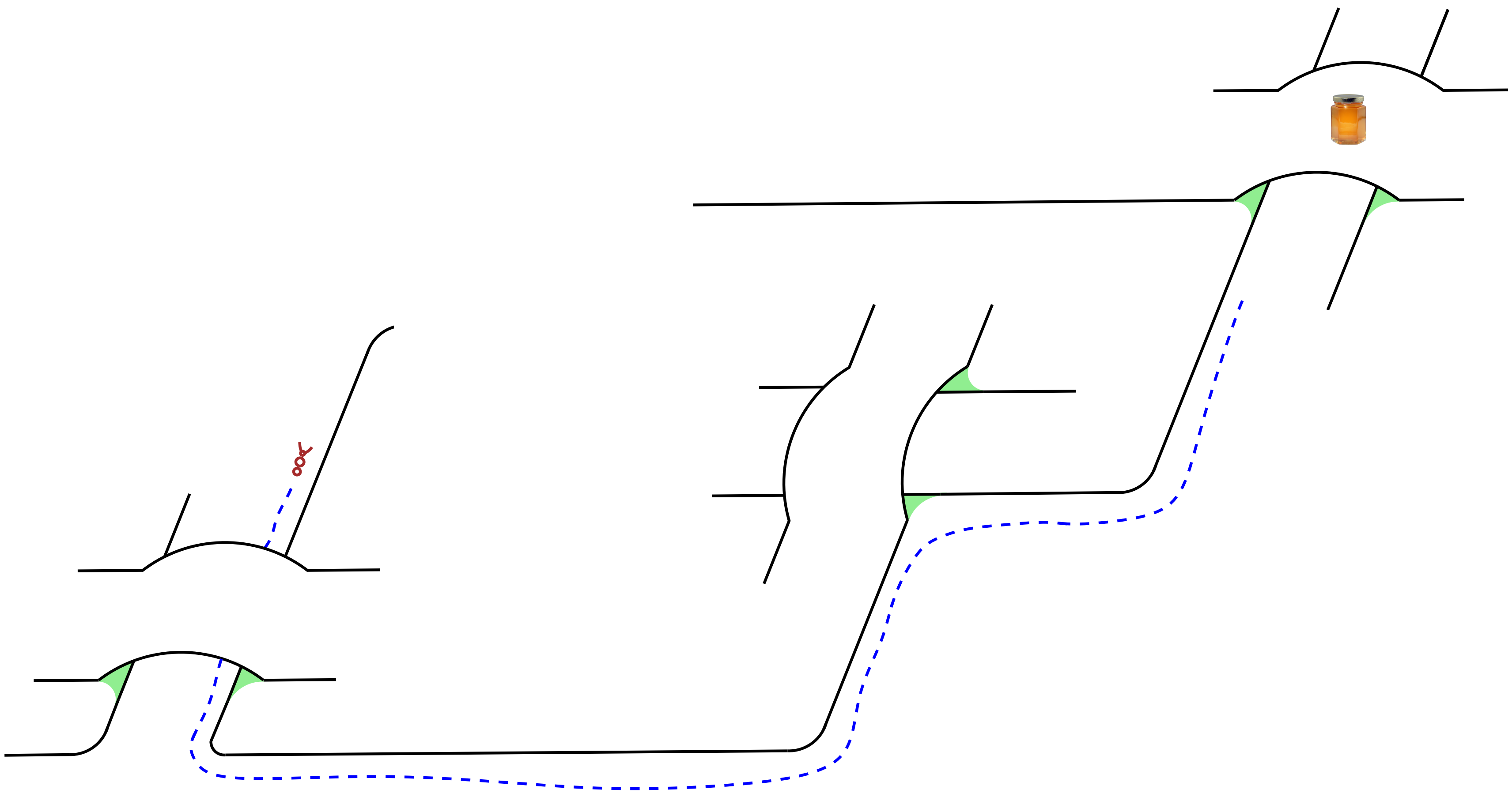
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

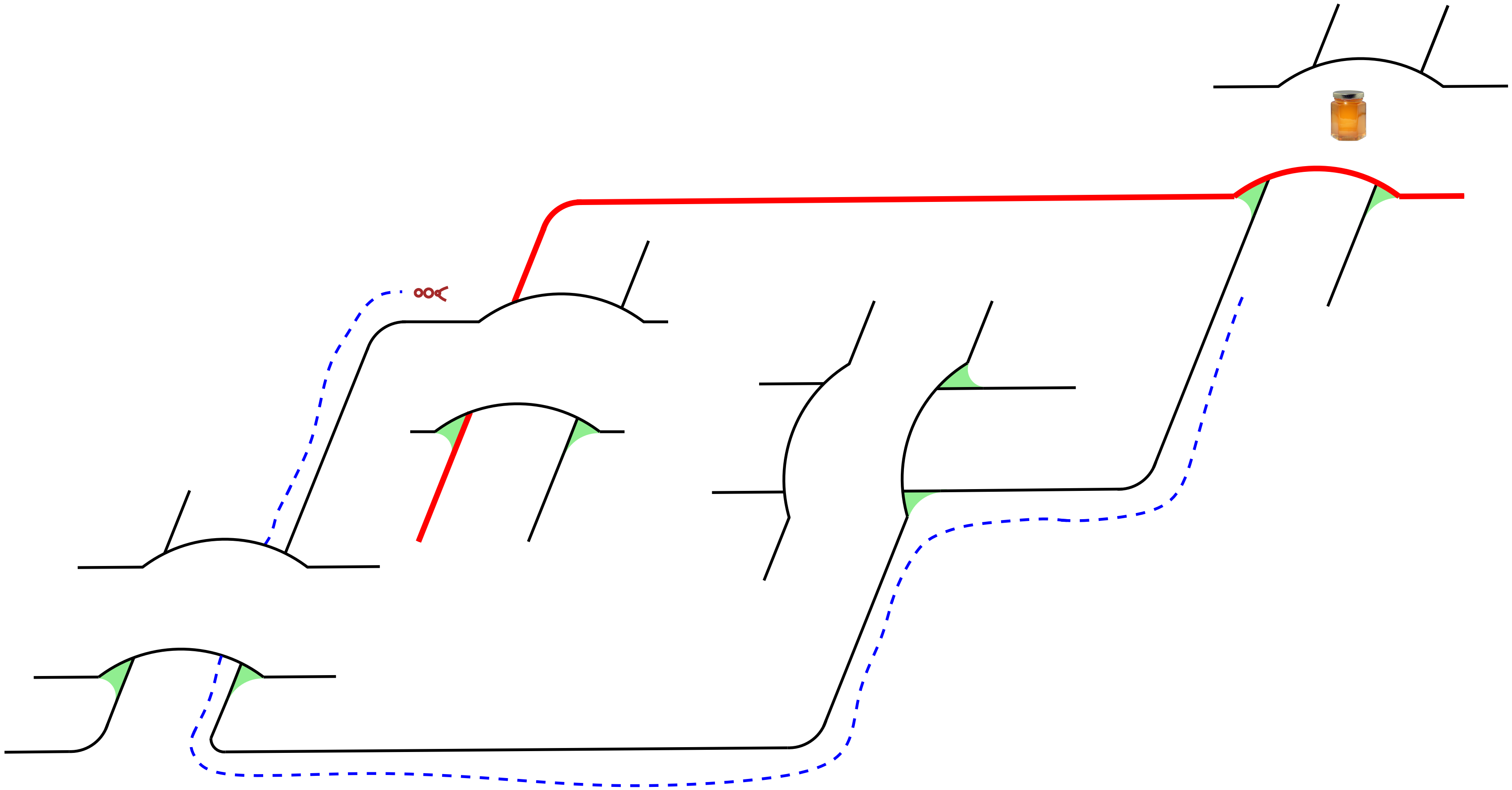
Proof.



# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

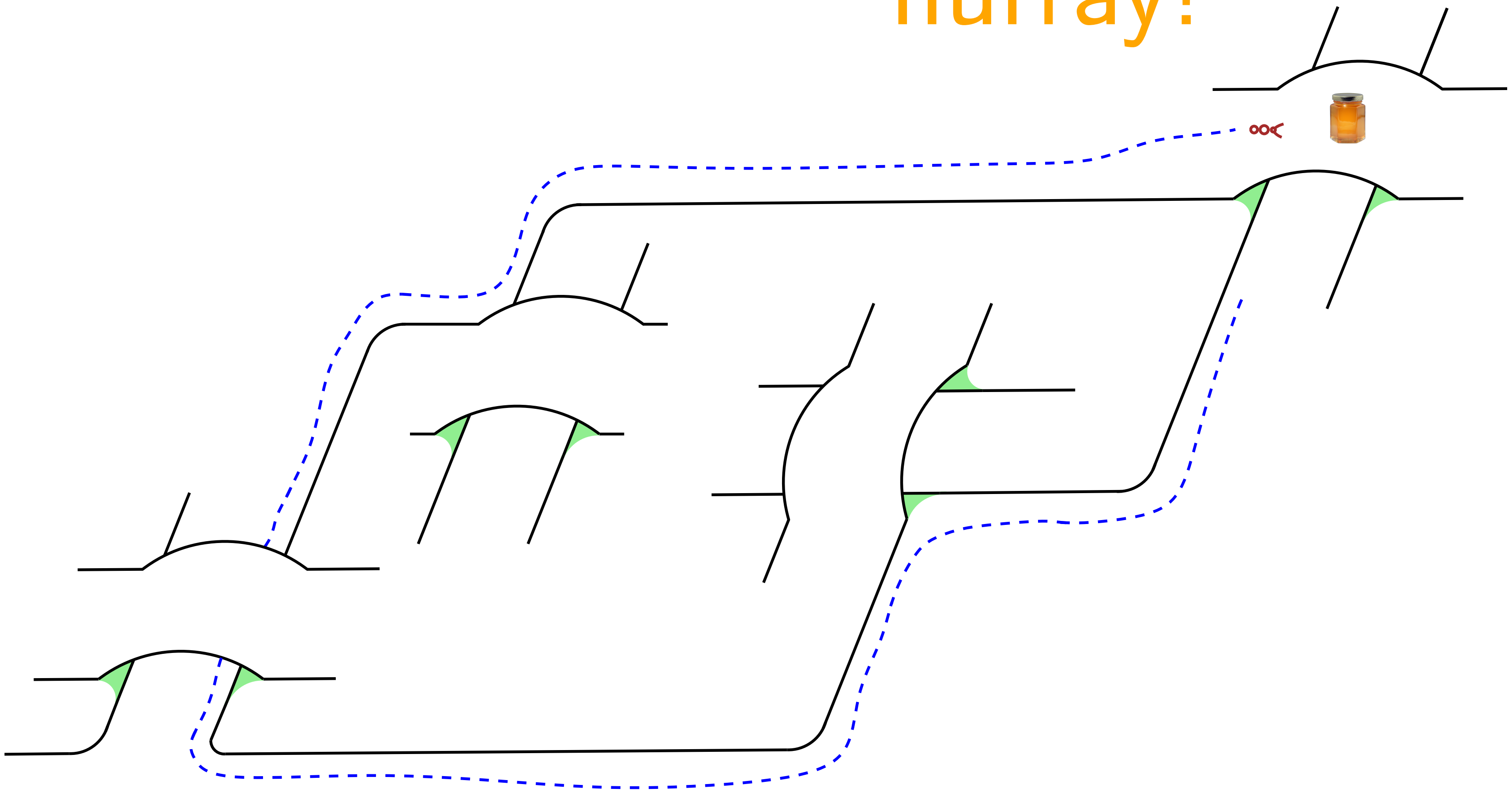


# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

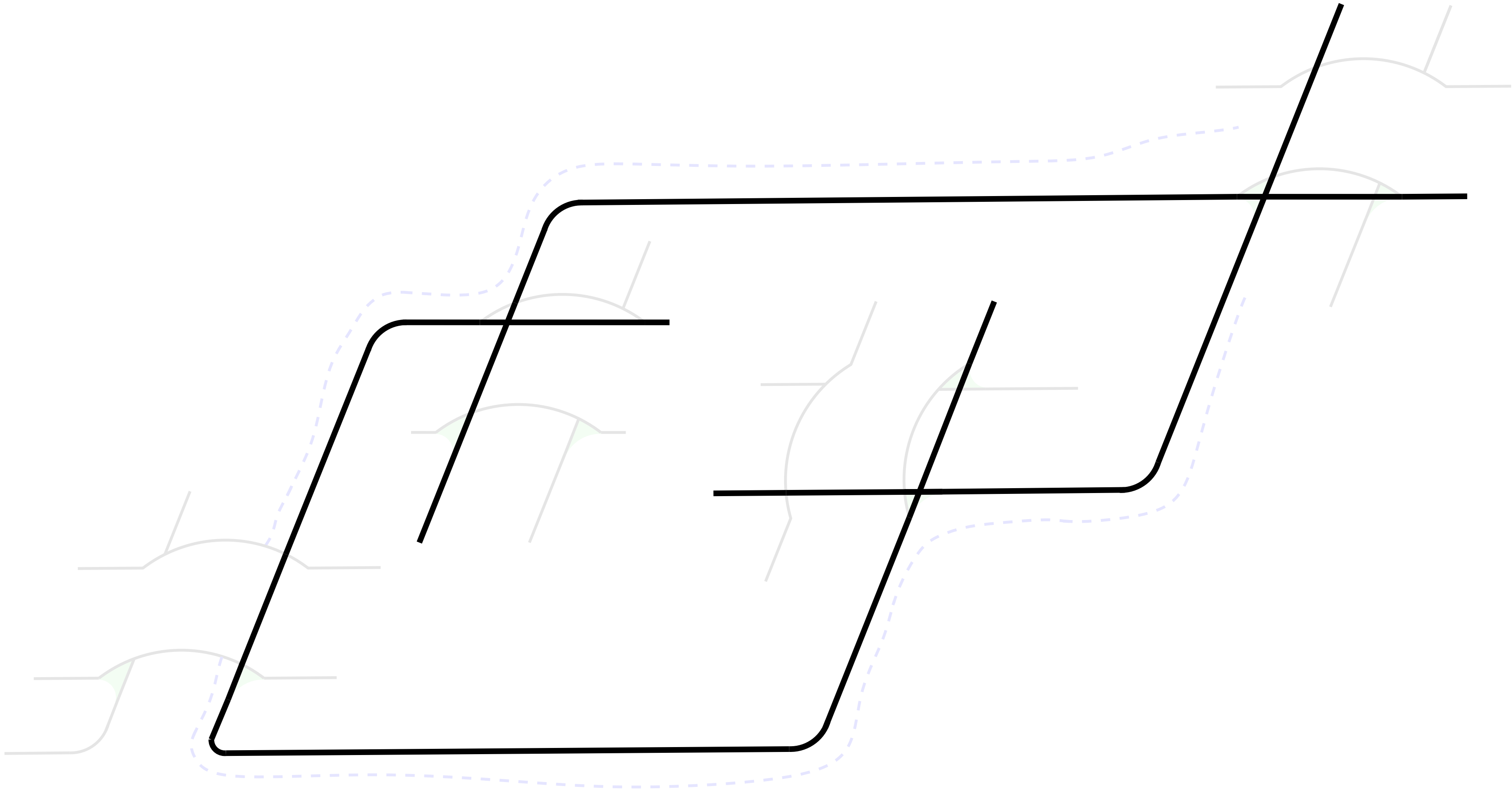
hurray!



# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.



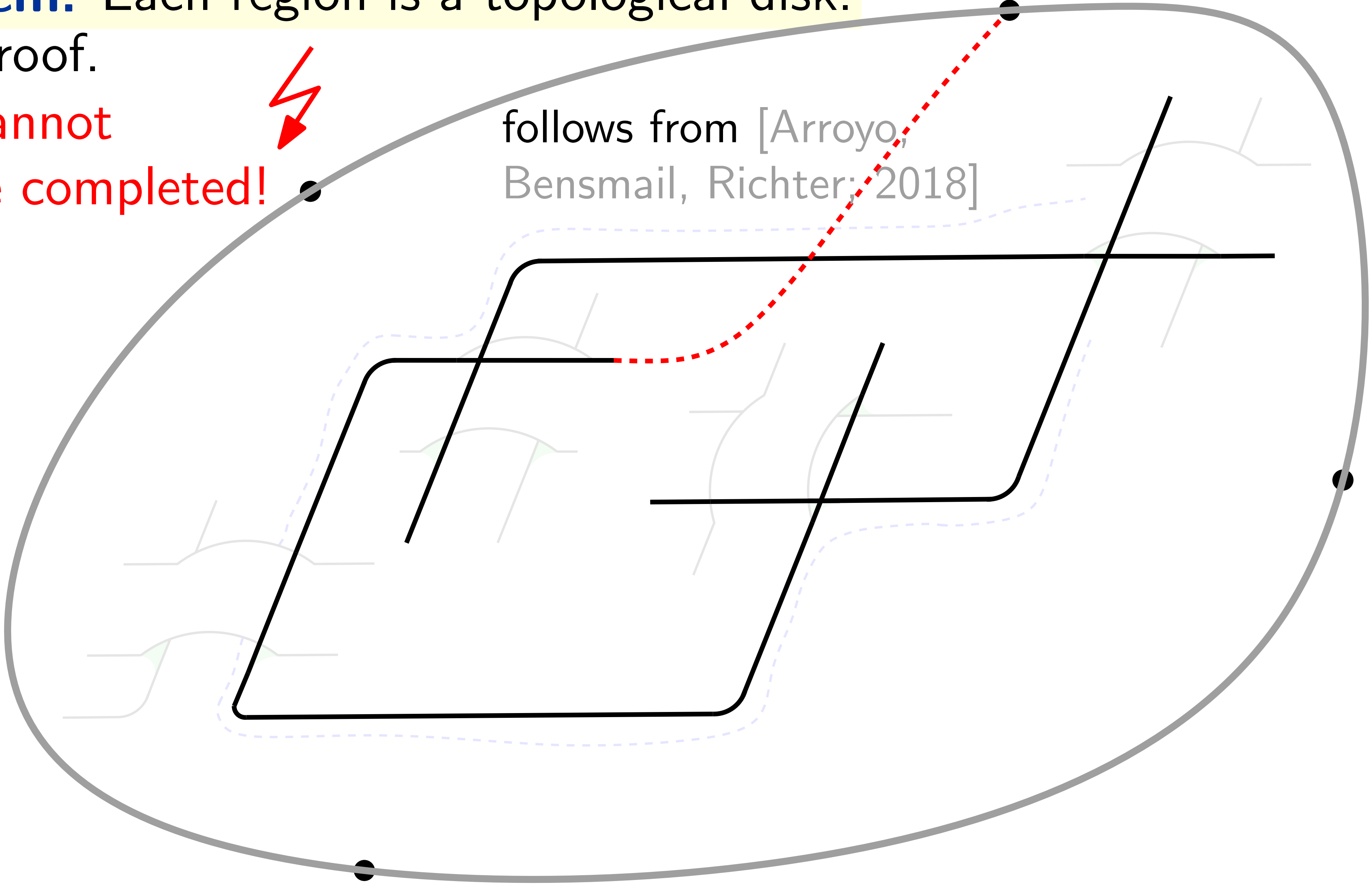
# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

Cannot
be completed!

follows from [Arroyo, Bensmail, Richter; 2018]



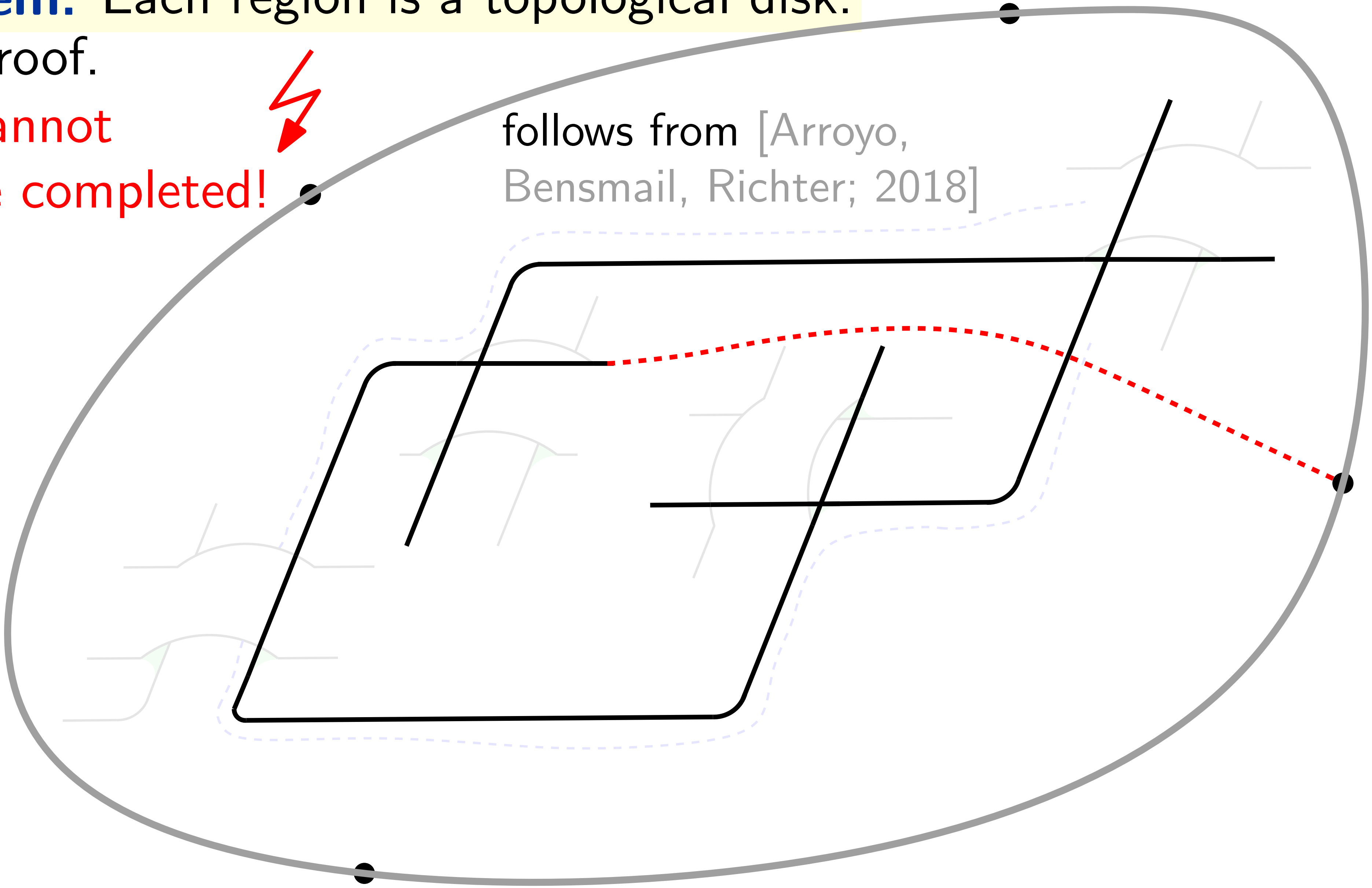
# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

Cannot
be completed!

follows from [Arroyo, Bensmail, Richter; 2018]

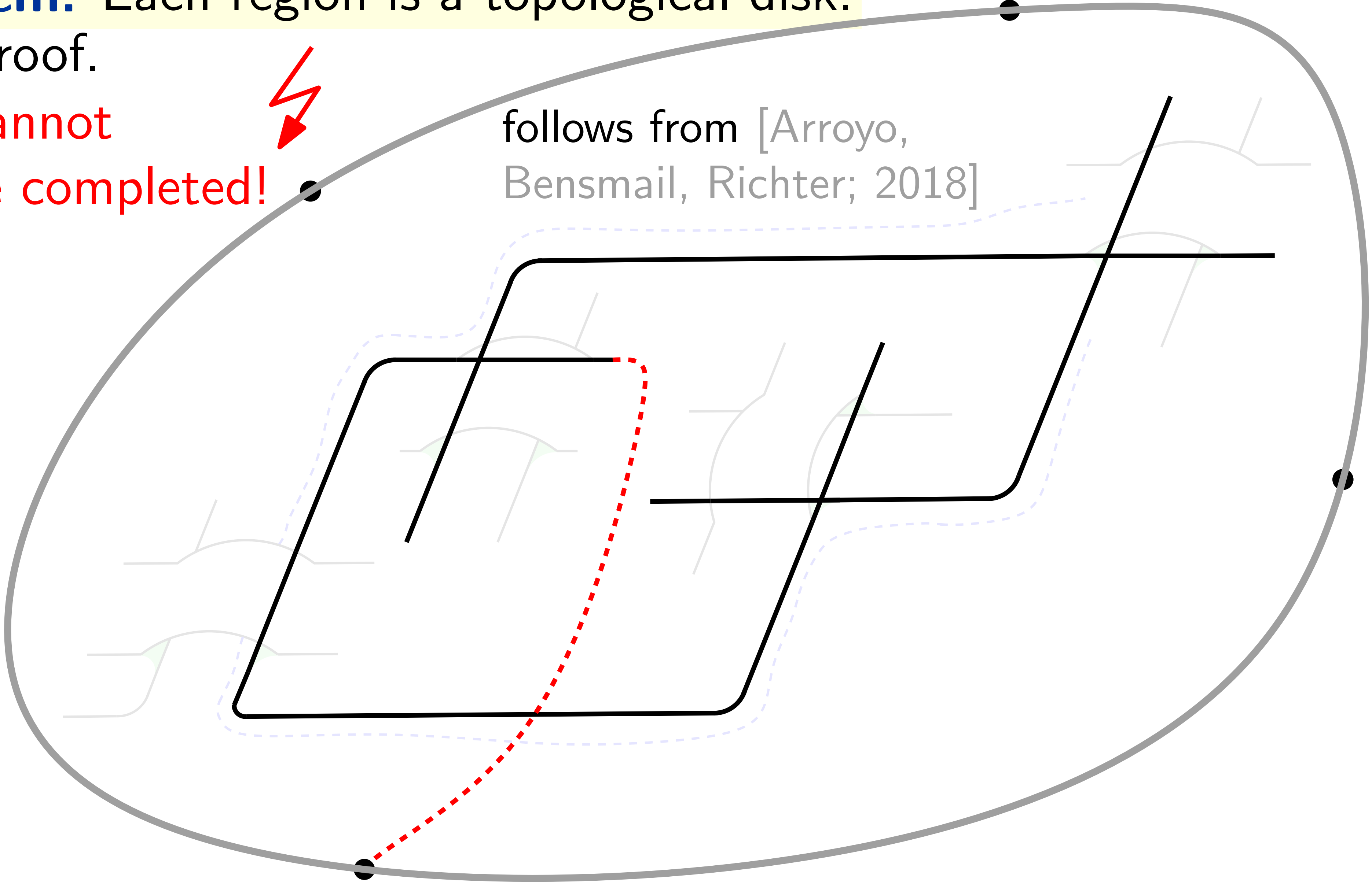


# A region is a topological disk

**Lem.** Each region is a topological disk.

Proof.

Cannot be completed!



# Open Questions

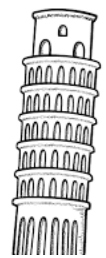
We have provided an FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$.

Since our algorithm is based on $\mathrm{MSO}_2$ the runtime is

# Open Questions

We have provided an FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$.

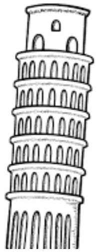
Since our algorithm is based on $\mathrm{MSO}_2$ the runtime is

 $(k) \times (|V| + |E|)$

# Open Questions

We have provided an FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$.

Since our algorithm is based on $\mathrm{MSO}_2$ the runtime is

 $(k)\times(|V|+|E|)$

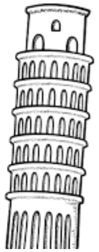
**Question 1**

Is there a faster FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$?

# Open Questions

We have provided an FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$.

Since our algorithm is based on $\mathrm{MSO}_2$ the runtime is



$$(k) \times (|V| + |E|)$$

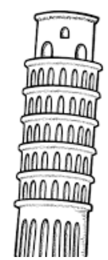
**Question 1**
Is there a faster FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$?

**Question 2**
Is deciding whether $\mathrm{bc}^{\circ}(G) = k$ NP-hard?

# Open Questions

We have provided an FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$.

Since our algorithm is based on $\mathrm{MSO}_2$ the runtime is

 $(k)\times(|V| + |E|)$

**Question 1**
Is there a faster FPT algorithm for deciding whether $\mathrm{bc}^{\circ}(G) = k$?

**Question 2**
Is deciding whether $\mathrm{bc}^{\circ}(G) = k$ NP-hard?

**Question 3**
Is bundle crossing min. also FPT for general simple layouts?