Fixed-Point Definability and Polynomial Time

Martin Grohe

Department of Computer Science
Humboldt University Berlin

Outline

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

Fagin's Theorem and Gurevich's Question

Fagin's Theorem (1974)

Existential second-order logic Σ^1_1 *captures* NP*.*

Fagin's Theorem and Gurevich's Question

Fagin's Theorem (1974)

Existential second-order logic Σ^1_1 *captures* NP*.*

That is, a property of finite structures is definable in Σ^1_1 *iff it is decidable by a nondeterministic polynomial time algorithm.*

Fagin's Theorem and Gurevich's Question

Fagin's Theorem (1974)

Existential second-order logic Σ^1_1 *captures* NP*.*

That is, a property of finite structures is definable in Σ^1_1 *iff it is decidable by a nondeterministic polynomial time algorithm.*

Question (Gurevich 1988)

Is there a logic that captures PTIME *?*

Fagin's Theorem and Gurevich's Question

Fagin's Theorem (1974)

Existential second-order logic Σ^1_1 *captures* NP*.*

That is, a property of finite structures is definable in Σ^1_1 *iff it is decidable by a nondeterministic polynomial time algorithm.*

Question (Gurevich 1988)

Is there a logic that captures PTIME *?*

Corollary (of Fagin's Theorem)

If no logic captures PTIME*, then* PTIME $\neq$ NP.

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely those queries that are computable in polynomial time are expressible ?

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely those queries that are computable in polynomial time are expressible ?

“Is there a query language that expresses precisely those queries that can be answered efficiently ?”

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely those queries that are computable in polynomial time are expressible ?

"Is there a query language that expresses precisely those queries that can be answered efficiently ?"

Observation

Chandra and Harel's question is equivalent to Gurevich's question for a logic capturing PTIME*:*

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely those queries that are computable in polynomial time are expressible ?

"Is there a query language that expresses precisely those queries that can be answered efficiently ?"

Observation

Chandra and Harel's question is equivalent to Gurevich's question for a logic capturing PTIME*:*

$$\begin{aligned} \textit{relational database} &\approx \textit{finite structure} \\ \textit{query} &\approx \textit{property of finite structures} \end{aligned}$$

Properties of Structures

Structures are finite and relational

Example

Graphs, Boolean circuits, finite words

Properties of Structures

Structures are finite and relational

Example

Graphs, Boolean circuits, finite words

A property of finite σ-structures

Properties of Structures

Structures are finite and relational

Example

Graphs, Boolean circuits, finite words

A property of finite σ-structures (aka Boolean query of vocabulary σ)

Properties of Structures

Structures are finite and relational

Example

Graphs, Boolean circuits, finite words

A property of finite σ-structures (aka Boolean query of vocabulary σ) is a class of structures of vocabulary σ that is closed under isomorphism.

Properties of Structures

Structures are finite and relational

Example

Graphs, Boolean circuits, finite words

A property of finite σ-structures (aka Boolean query of vocabulary σ) is a class of structures of vocabulary σ that is closed under isomorphism.

Example

Connectedness of graphs, satisfiability of Boolean circuits, the language $\{a^n b^n \mid n \geq 0\}$

Abstract Logics

A logic L consists of a set of sentences and a semantics that associates a property $\mathcal{P}_\phi$ of finite structures with each sentence ϕ.

A structure A satisfies of a sentence ϕ if $A \in \mathcal{P}_\phi$.

Abstract Logics

A logic L consists of a set of sentences and a semantics that associates a property $\mathcal{P}_\phi$ of finite structures with each sentence ϕ.

A structure A satisfies of a sentence ϕ if $A \in \mathcal{P}_\phi$.

A property $\mathcal{P}$ is definable in L if $\mathcal{P} = \mathcal{P}_\phi$ for some sentence ϕ of L.

Abstract Logics

A logic L consists of a set of sentences and a semantics that associates a property $\mathcal{P}_\phi$ of finite structures with each sentence ϕ.

A structure A satisfies of a sentence ϕ if $A \in \mathcal{P}_\phi$.

A property $\mathcal{P}$ is definable in L if $\mathcal{P} = \mathcal{P}_\phi$ for some sentence ϕ of L.

Remark

In addition to the minimal requirement of an isomorphism invariant semantics, logics are usually required to satisfy various regularity conditions and effectiveness conditions.

Logics capturing PTIME

Working Definition

A logic L captures PTIME
if for every property $\mathcal{P}$ of finite structures:

$$\mathcal{P} \text{ definable in L} \iff \mathcal{P} \text{ decidable in PTIME}.$$

Logics capturing PTIME

Working Definition

A logic L captures PTIME
if for every property $\mathcal{P}$ of finite structures:

$$\mathcal{P} \text{ definable in L} \iff \mathcal{P} \text{ decidable in PTIME}.$$

Remark

The actual definition is more subtle and requires various effectiveness conditions to exclude pathological examples.

Logics capturing PTIME

Working Definition

A logic L captures PTIME on a class $\mathcal{C}$ of structures
if for every property $\mathcal{P}$ of finite structures:

$$\mathcal{P} \text{ definable in L on } \mathcal{C} \iff \mathcal{P} \text{ decidable in PTIME on } \mathcal{C}.$$

Remark

The actual definition is more subtle and requires various effectiveness conditions to exclude pathological examples.

Representation Invariance

Main technical difficulty

Representation Invariance

Main technical difficulty

- Machines operate on string encodings of structures

Representation Invariance

Main technical difficulty

- Machines operate on string encodings of structures
- Logics operate directly on structures

Representation Invariance

Main technical difficulty

- Machines operate on string encodings of structures
- Logics operate directly on structures
- Structures have no (efficiently computable) canonical string encoding

Representation Invariance

Main technical difficulty

- Machines operate on string encodings of structures
- Logics operate directly on structures
- Structures have no (efficiently computable) canonical string encoding

Remark

In NP we can “guess” a string encoding, that’s why this problem disappears for complexity classes that contain NP.

Representation Invariance

Main technical difficulty

- Machines operate on string encodings of structures
- Logics operate directly on structures
- Structures have no (efficiently computable) canonical string encoding

Remark

In NP we can “guess” a string encoding, that’s why this problem disappears for complexity classes that contain NP.

Canonization

If there is a polynomial time algorithm that computes a canonical representation for a given structure from a class $\mathcal{C}$, then there is a logic that captures PTIME on $\mathcal{C}$.

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Ordered Structures

- Distinguished binary relation symbol $\leq$.

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Ordered Structures

- Distinguished binary relation symbol $\leq$.
- Ordered structure: vocabulary contains $\leq$, interpreted by a linear order of the universe

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Ordered Structures

- Distinguished binary relation symbol $\leq$.
- Ordered structure: vocabulary contains $\leq$, interpreted by a linear order of the universe
- $\mathcal{O}$ class of all ordered structures.

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Ordered Structures

- Distinguished binary relation symbol $\leq$.
- Ordered structure: vocabulary contains $\leq$, interpreted by a linear order of the universe
- $\mathcal{O}$ class of all ordered structures.
- Ordered structures admit efficient canonization.

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Ordered Structures

- Distinguished binary relation symbol $\leq$.
- Ordered structure: vocabulary contains $\leq$, interpreted by a linear order of the universe
- $\mathcal{O}$ class of all ordered structures.
- Ordered structures admit efficient canonization.

Immerman-Vardi Theorem (1982)

FP *captures* PTIME *on* $\mathcal{O}$.

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

FP cannot count

Lemma (Folklore)

For every FP*-sentence* ϕ *there is a* k *such that for all structures* A, B *with* $|A|, |B| \geq k$ *and only empty relations:*

$$A \models \phi \iff B \models \phi.$$

FP cannot count

Lemma (Folklore)

For every FP*-sentence* ϕ *there is a* k *such that for all structures* A, B *with* $|A|, |B| \geq k$ *and only empty relations:*

$$A \models \phi \iff B \models \phi.$$

Proof.

1. Show that ϕ is equivalent to sentence ϕ' of k-variable infinitary logic, for a suitable k.

□

FP cannot count

Lemma (Folklore)

For every FP*-sentence* ϕ *there is a* k *such that for all structures* A, B *with* $|A|, |B| \geq k$ *and only empty relations:*

$$A \models \phi \iff B \models \phi.$$

Proof.

1. Show that ϕ is equivalent to sentence ϕ' of k-variable infinitary logic, for a suitable k.
2. Proof claim for ϕ' by k-pebble game argument.

□

FP cannot count

Lemma (Folklore)

For every FP*-sentence* ϕ *there is a* k *such that for all structures* A, B *with* $|A|, |B| \geq k$ *and only empty relations:*

$$A \models \phi \iff B \models \phi.$$

Corollary

FP *does not capture* PTIME*.*

FP cannot count

Lemma (Folklore)

For every FP*-sentence* ϕ *there is a* k *such that for all structures* A, B *with* $|A|, |B| \geq k$ *and only empty relations:*

$$A \models \phi \iff B \models \phi.$$

Corollary

FP *does not capture* PTIME*.*

Proof.

The query

"The universe has even cardinality"

is decidable in PTIME,
but not expressible in FP (by Lemma). □

The CFI-Theorem

Fixed-Point Logic with Counting:
FP+C = FP + Counting Operators

The CFI-Theorem

Fixed-Point Logic with Counting:
FP+C = FP + Counting Operators

Theorem (Cai, Fürer, Immerman 1992)

FP+C *does not capture* PTIME*.*

Capturing PTIME Almost Everywhere

. . .

Theorem (Hella, Kolaitis, Luosto 1996)

FP+C *captures* PTIME *almost everywhere*.

. . . and Even Somewhere Interesting

Theorem

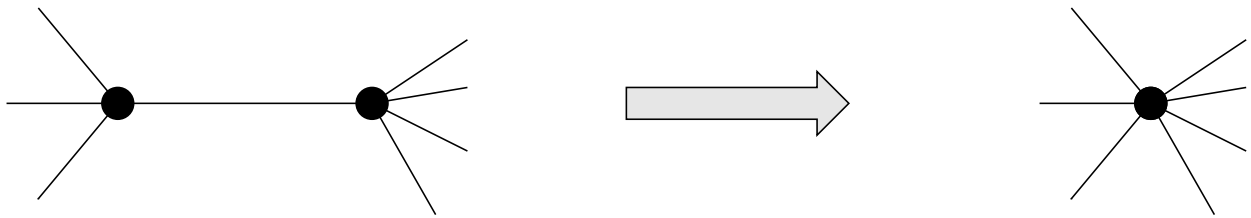
FP+C *captures* PTIME *on the following classes of graphs:*

1. *Trees (Immerman, Lander 1990)*
2. *Classes of graphs of bounded treewidth (G., Mariño 1999)*
3. *Planar graphs (G. 1998)*
4. *Chordal Line Graphs (G. 2009)*

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

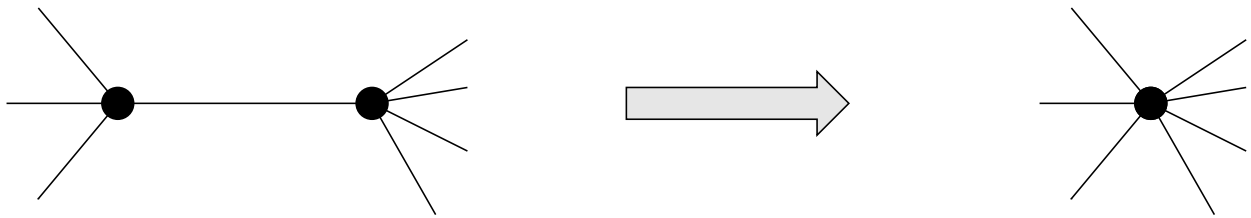
Minors and Classes Excluding Minors

Graph *G* is a minor of graph *H*
if *G* is obtained from *H* by deleting edges and vertices and contracting edges.



Minors and Classes Excluding Minors

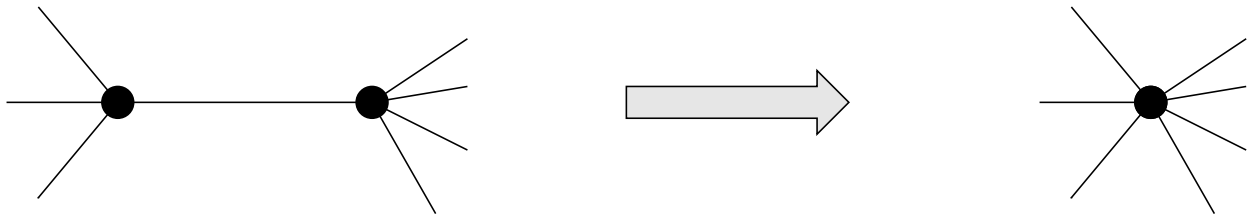
Graph G is a minor of graph H
if G is obtained from H by deleting edges and vertices and contracting edges.



H is an excluded minor for class $\mathcal{C}$ of graphs
if $H \not\preceq G$ for any $G \in \mathcal{C}$

Minors and Classes Excluding Minors

Graph G is a minor of graph H
if G is obtained from H by deleting edges and vertices and contracting edges.



H is an excluded minor for class $\mathcal{C}$ of graphs
if $H \not\preceq G$ for any $G \in \mathcal{C}$

Example

Classes of bounded treewidth, planar graphs, graphs of bounded genus all exclude minors

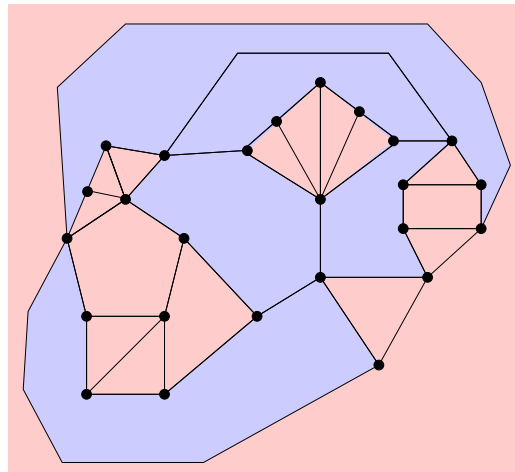
Capturing PTIME on Classes with Excluded Minors

Main Theorem

Let $\mathcal{C}$ be a class of graphs that excludes a minor.
Then FP+C *captures* PTIME *on* $\mathcal{C}$*.*

Proof of the Main Theorem — Step 1:

Definable Orders and 3-Connected Planar Graphs



Definable orders

Lemma

If structures in class $\mathcal{C}$ admit FP*-definable order (possibly with parameters) then* FP *captures* PTIME *on $\mathcal{C}$.*

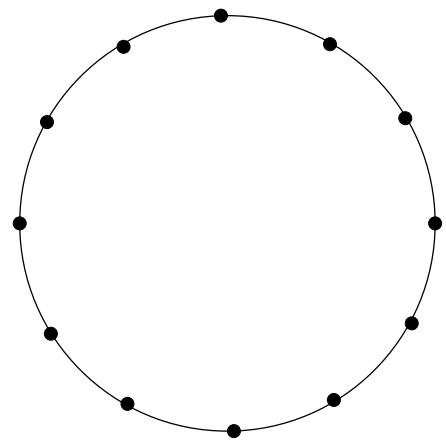
Definable orders

Lemma

If structures in class $\mathcal{C}$ admit FP*-definable order (possibly with parameters) then* FP *captures* PTIME *on $\mathcal{C}$.*

Example

Cycles admit order definable by FP-formula with 2 parameters.



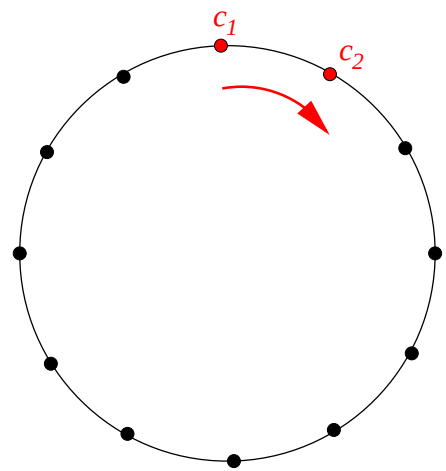
Definable orders

Lemma

If structures in class $\mathcal{C}$ admit FP-definable order (possibly with parameters) then FP captures PTIME on $\mathcal{C}$.

Example

Cycles admit order definable by FP-formula with 2 parameters.



Capturing PTIME on 3-Connected Planar Graphs

Lemma (G. 1998)

3-connected planar graphs admit an FP*-definable order.*

Corollary

FP *captures* PTIME *on the class of 3-connected planar graphs.*

Facial Cycles

Facts

1. *Every planar graph of minimum degree at least 3 has a facial cycle of length at most* 6 *(Folklore, based on Euler's formula)*.

Facial Cycles

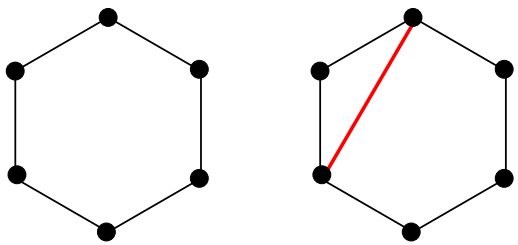
Facts

1. *Every planar graph of minimum degree at least 3 has a facial cycle of length at most* 6 *(Folklore, based on Euler's formula).*
2. *All planar embeddings of a 3-connected planar graph have the same facial cycles, and these are precisely the chordless and non-separating cycles (Whitney).*

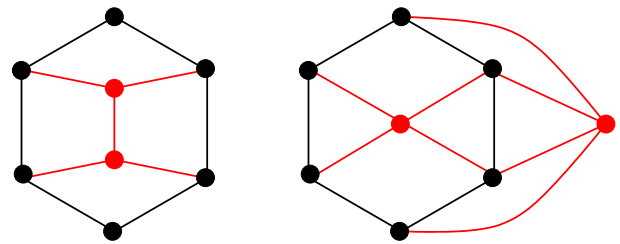
Facial Cycles

Facts

1. *Every planar graph of minimum degree at least 3 has a facial cycle of length at most* 6 *(Folklore, based on Euler's formula).*
2. *All planar embeddings of a 3-connected planar graph have the same facial cycles, and these are precisely the chordless and non-separating cycles (Whitney).*

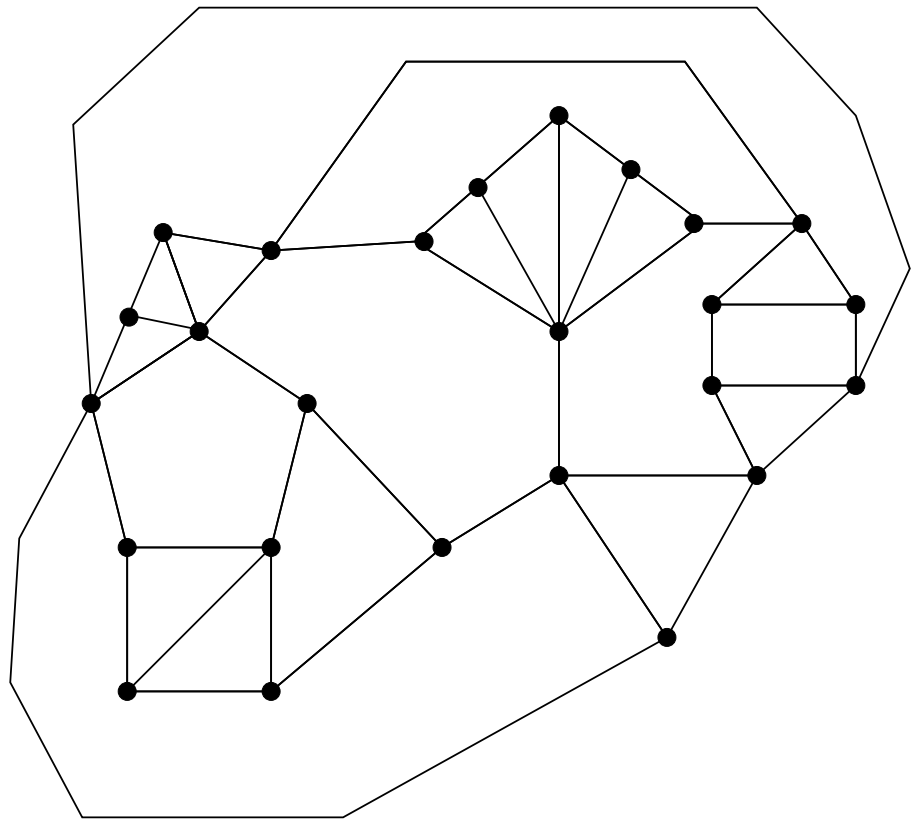


A chordless cycle and a cycle with a chord



A nonseparating and a separating cycle

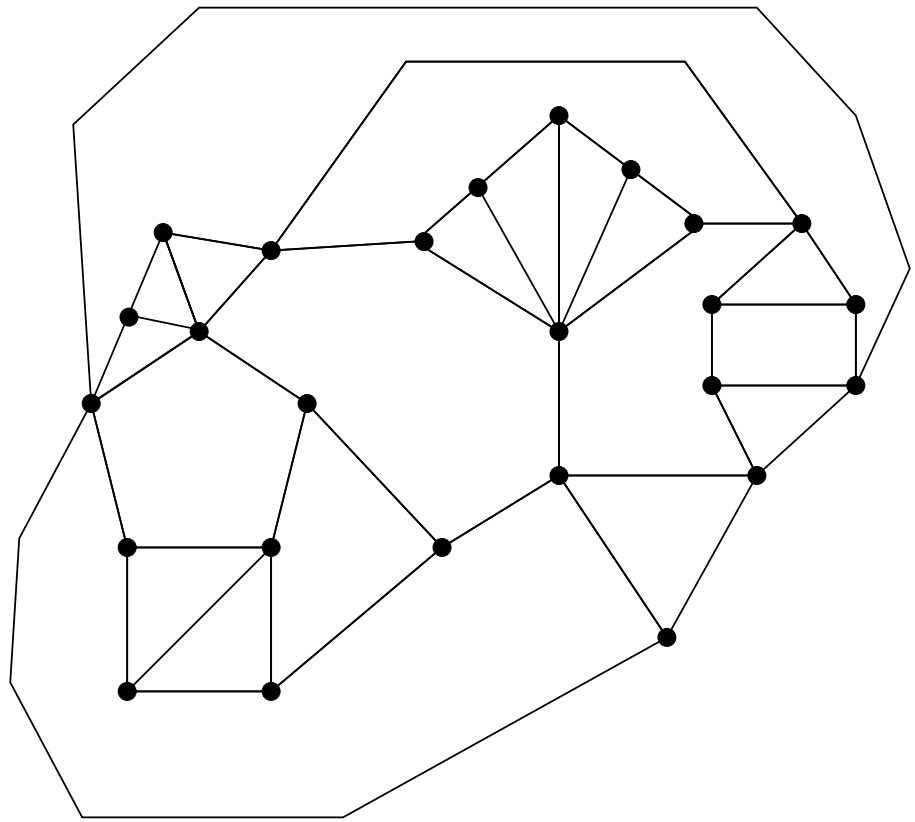
Defining an Order



Defining an Order

Idea

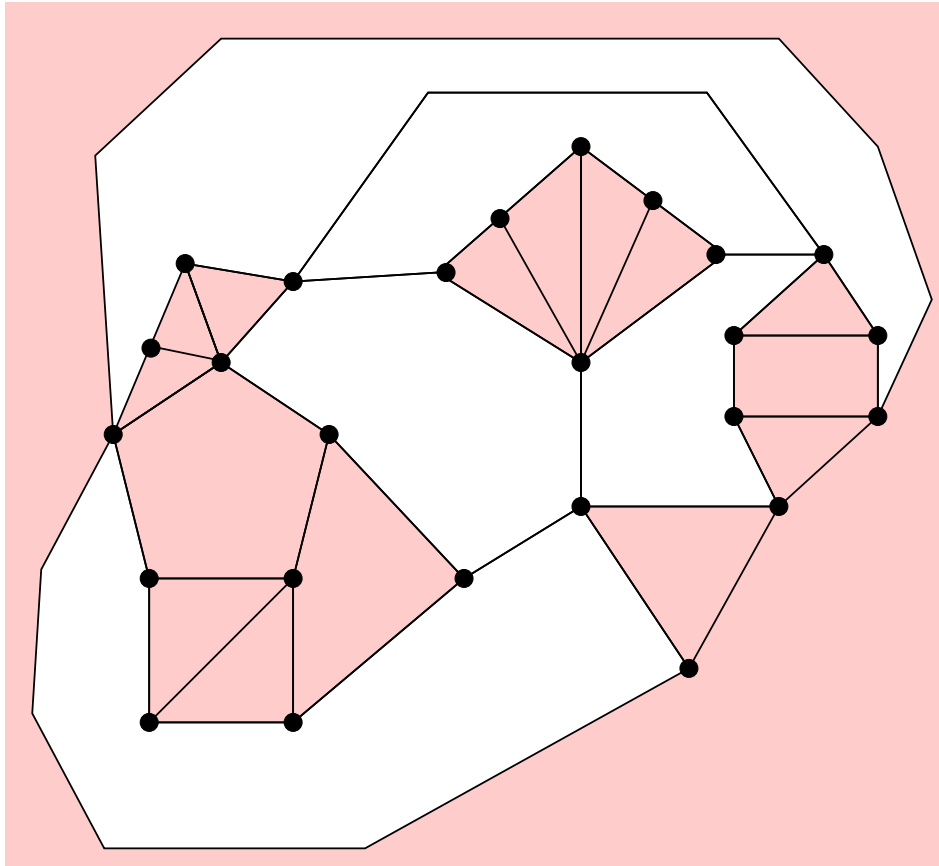
1. Define set of facial cycles



Defining an Order

Idea

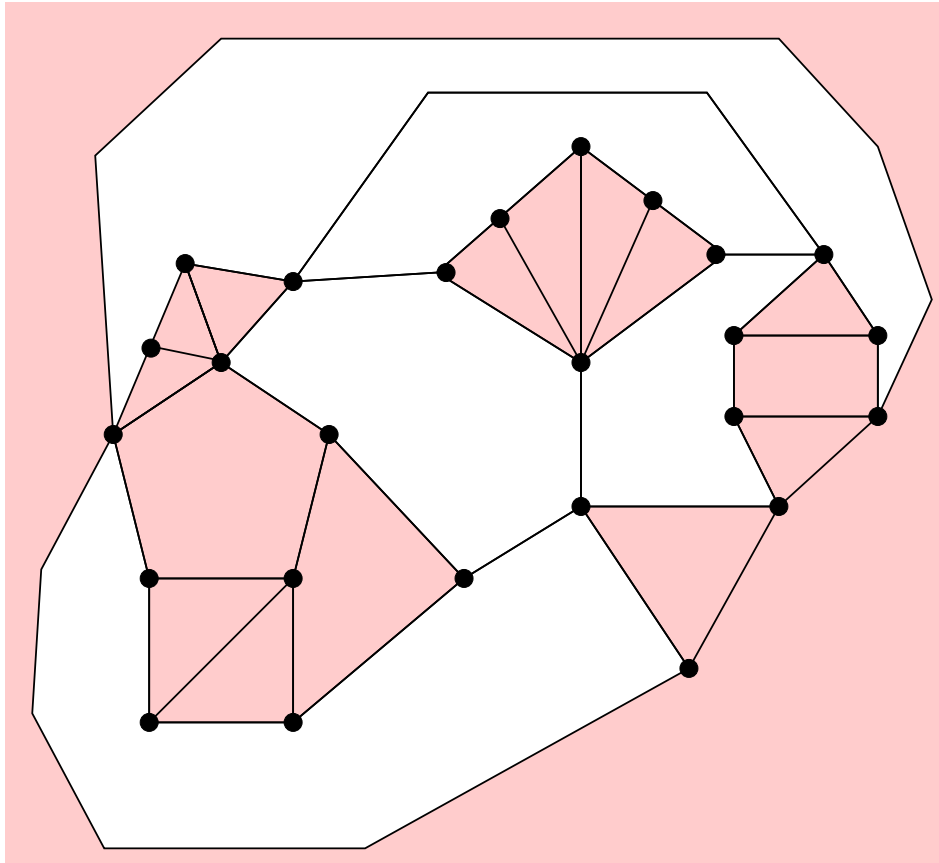
1. Define set of facial cycles
   - define facial cycles of length ≤ 6



Defining an Order

Idea

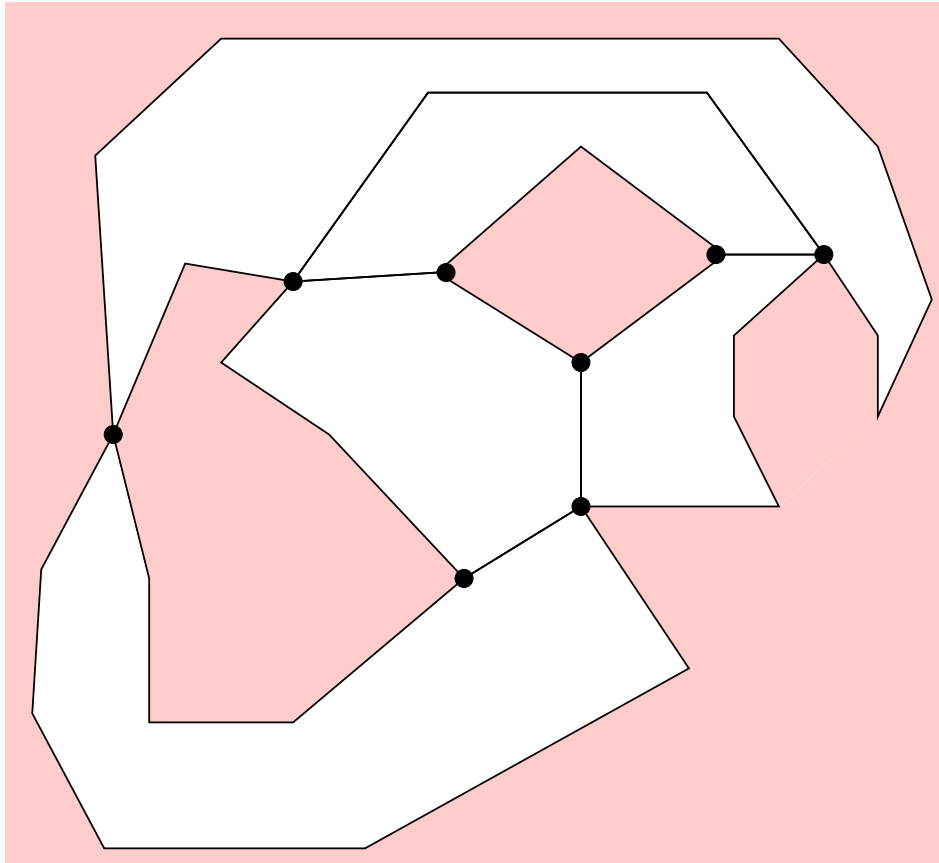
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

Idea

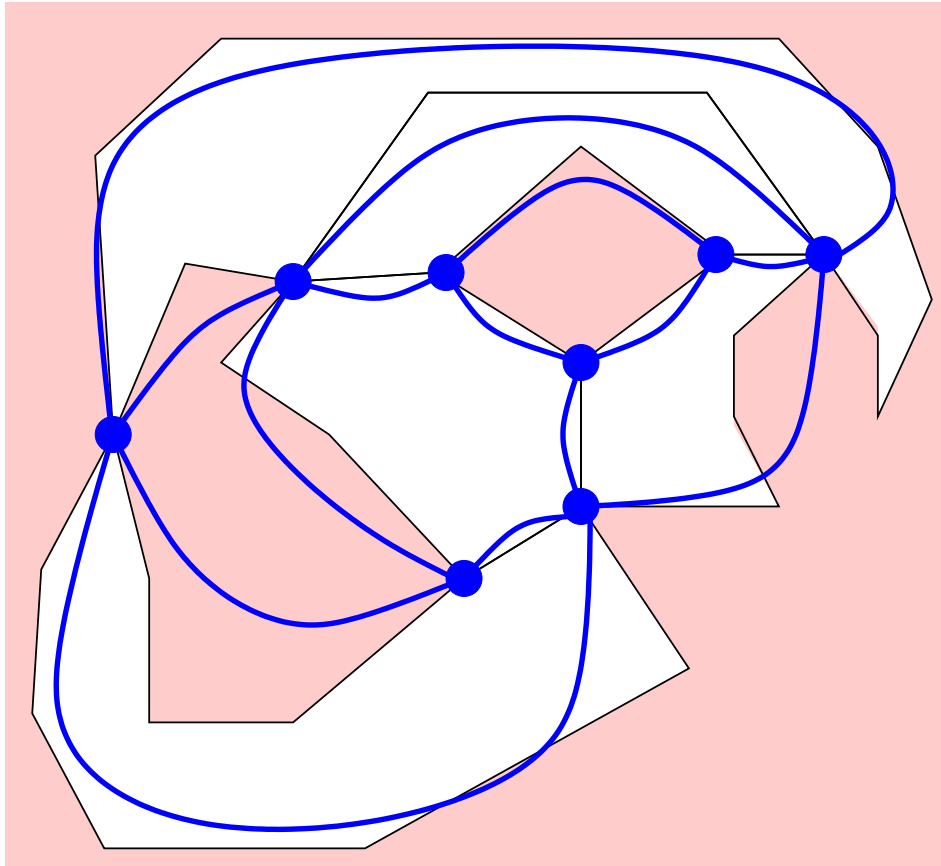
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

Idea

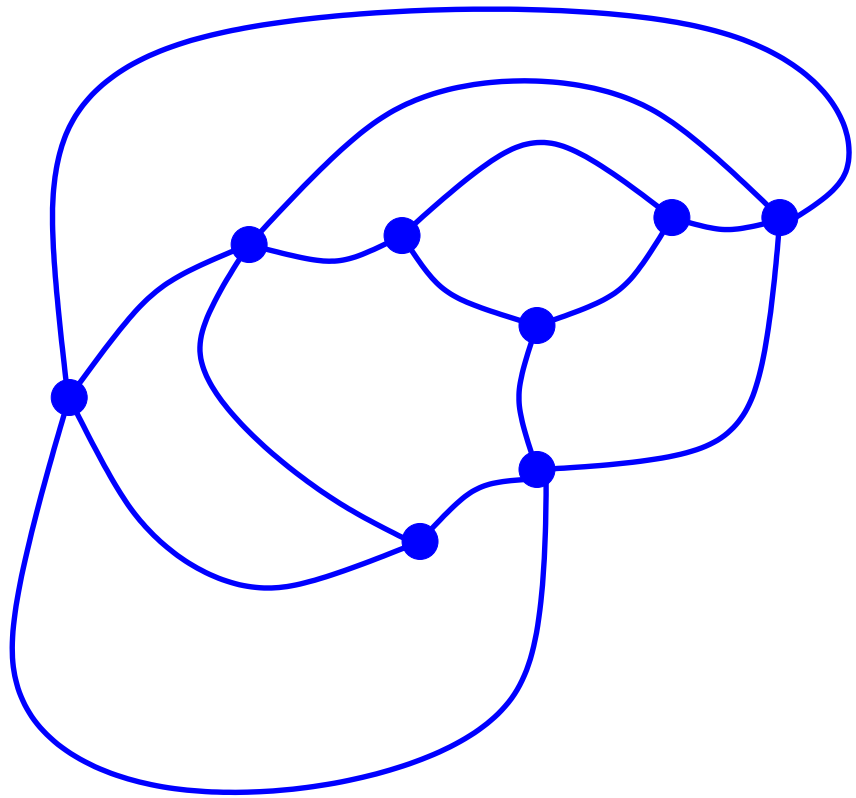
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

Idea

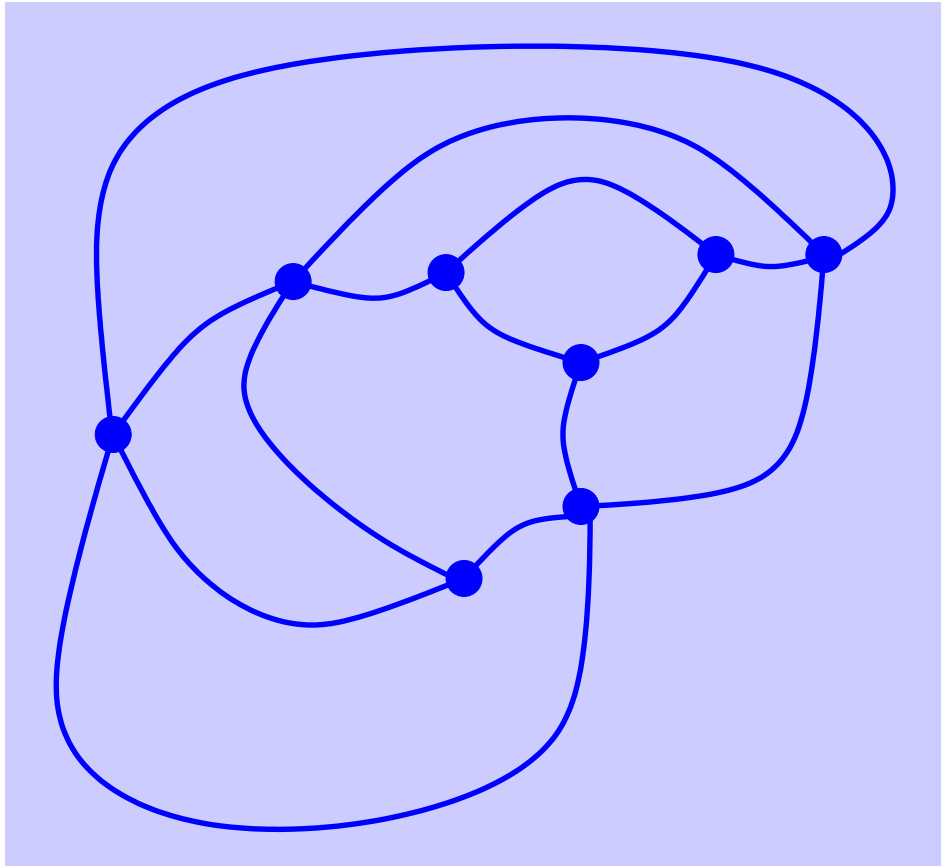
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

Idea

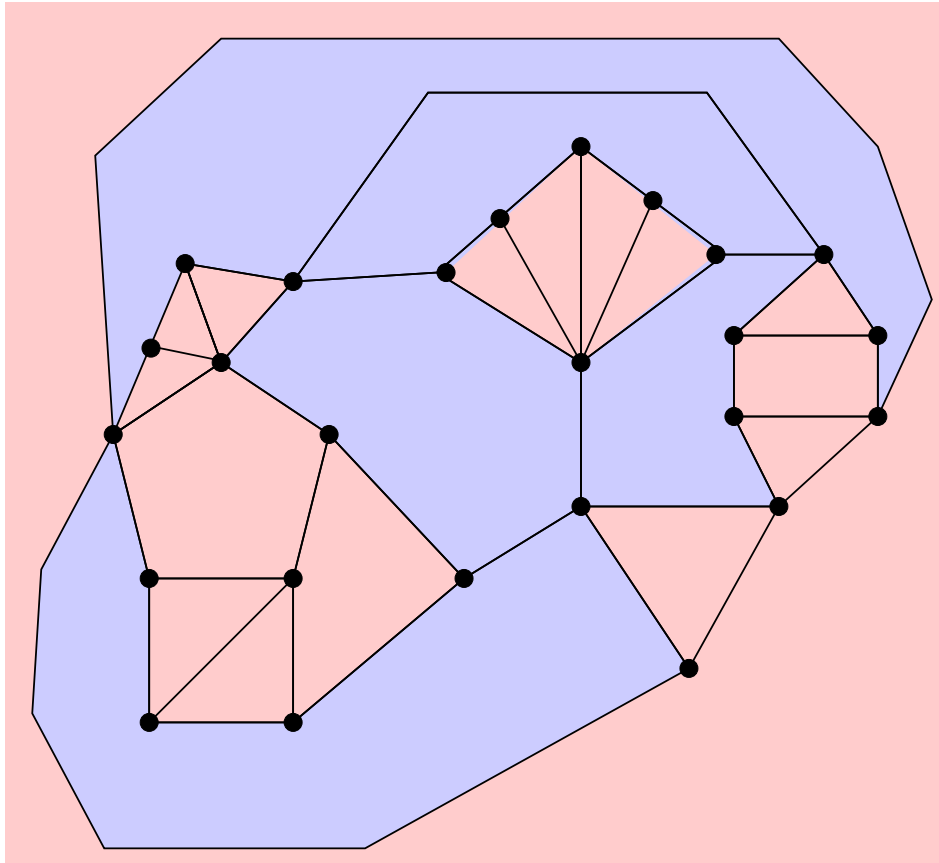
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

Idea

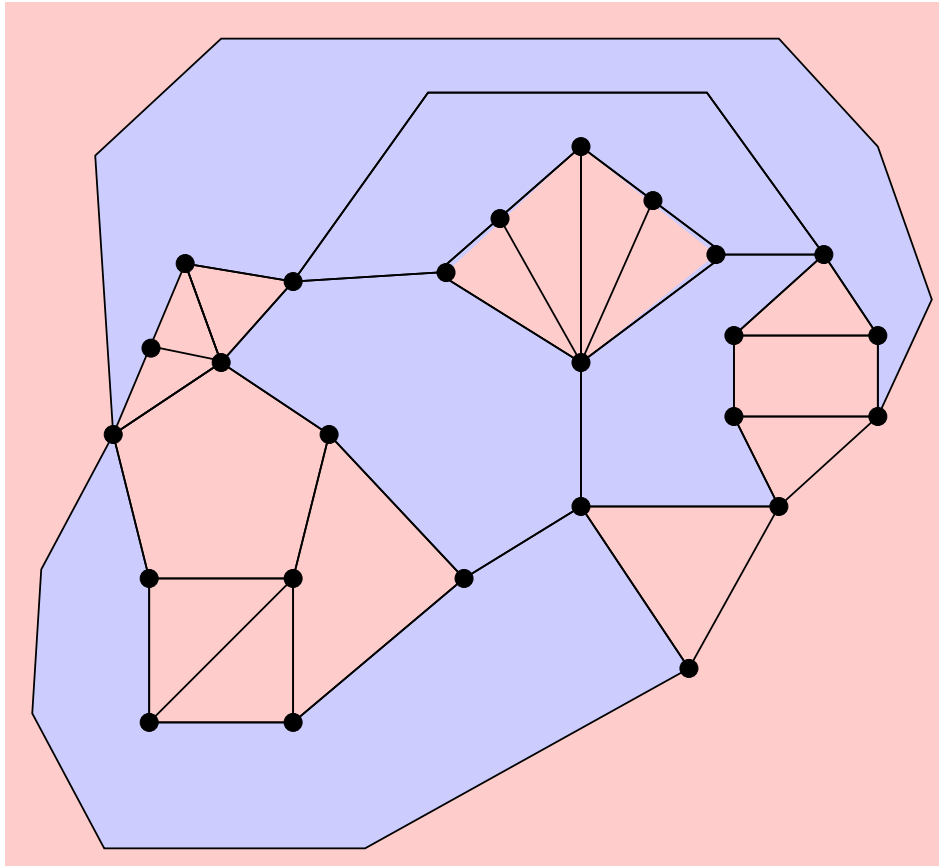
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat



Defining an Order

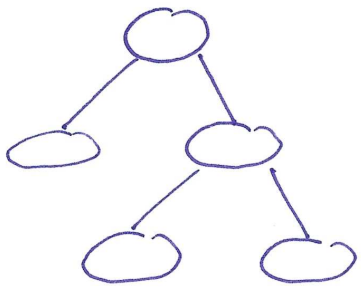
Idea

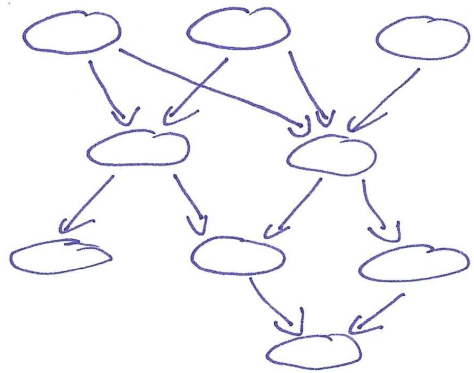
1. Define set of facial cycles
   - define facial cycles of length ≤ 6
   - simplify the graph and repeat
2. use facial cycles to define order



Proof of the Main Theorem — Step 2:

Ordered Treelike Decompositions and Arbitrary Planar Graphs

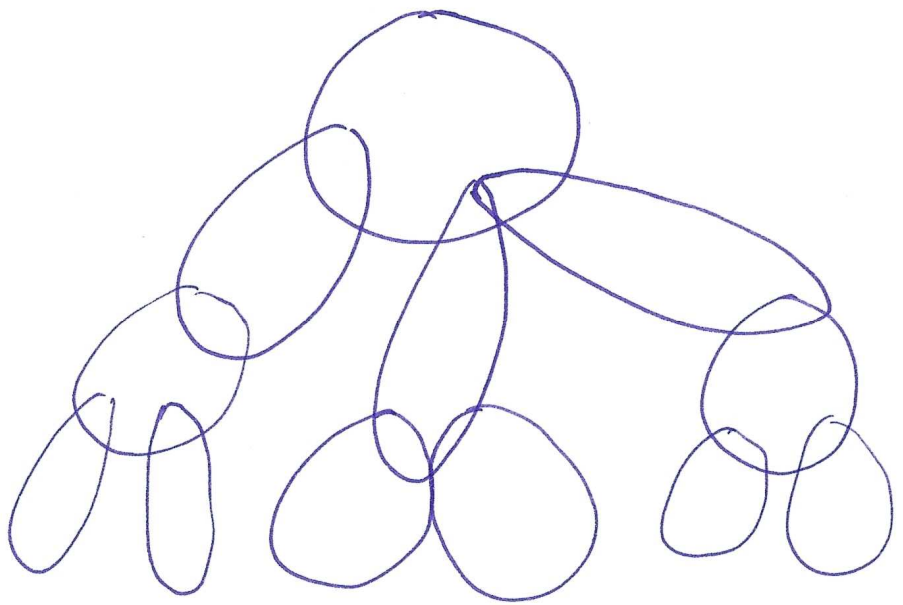




3-Connected Components

Fact

Every graph has a tree decomposition into 3-connected pieces.

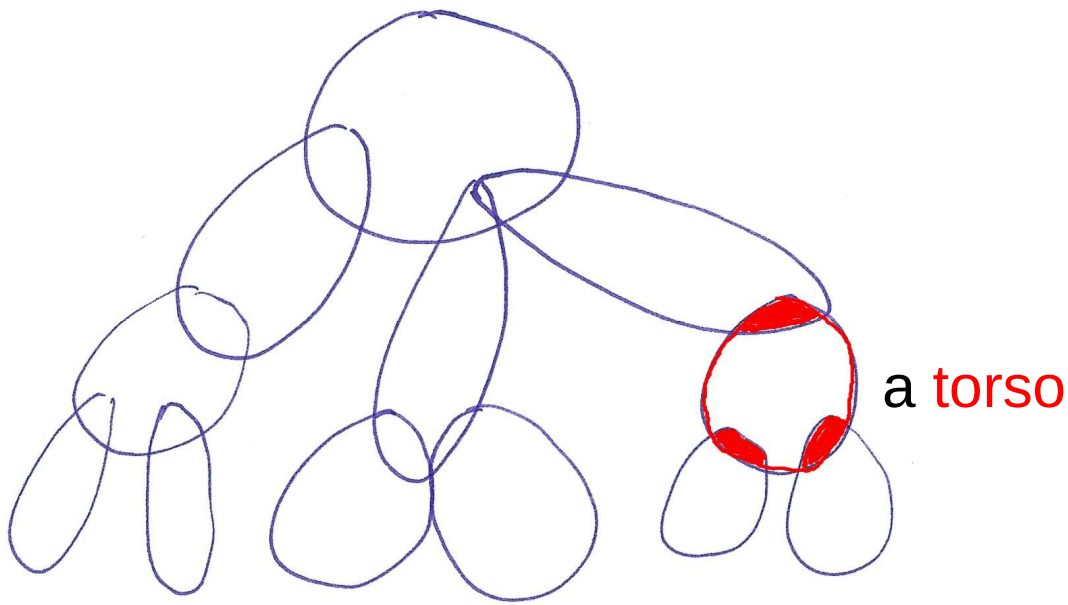


3-Connected Components

Fact

Every graph has a tree decomposition into 3-connected pieces.

The pieces of the decomposition (formally: torsi) are not necessarily subgraphs, but minors of the original graph.



Treelike Decomposition

Problem

Tree decompositions are not invariant under automorphisms and hence not definable.

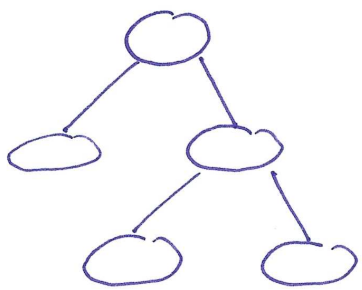
Treelike Decomposition

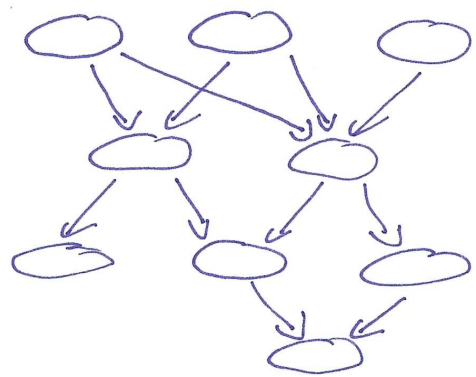
Problem

Tree decompositions are not invariant under automorphisms and hence not definable.

Solution

Use treelike decompositions whose underlying graph is not a tree but a directed acyclic graph.





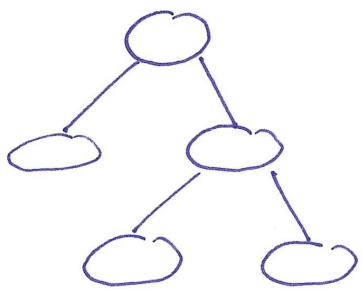
Treelike Decomposition

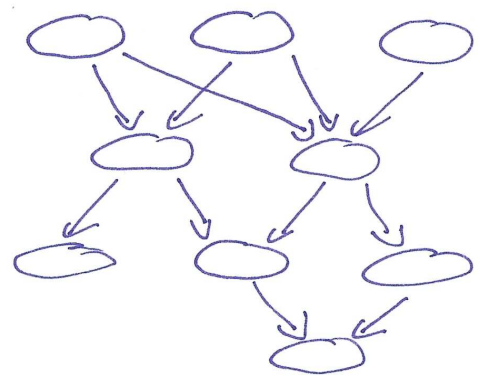
Problem

Tree decompositions are not invariant under automorphisms and hence not definable.

Solution

Use treelike decompositions whose underlying graph is not a tree but a directed acyclic graph.





Lemma

Every graph has an FP-definable treelike decomposition into 3-connected torsi that are minors of the graph.

Definable Ordered Treelike Decompositions

Definition

An FP-definable ordered treelike decomposition is an FP-definable treelike decomposition into torsi that admit an FP-definable linear order.

Definable Ordered Treelike Decompositions

Definition

An FP-definable ordered treelike decomposition is an FP-definable treelike decomposition into torsi that admit an FP-definable linear order.

Corollary

Planar graphs admit FP-definable ordered treelike decompositions.

Capturing PTIME on Planar Graphs

Lemma

Let $\mathcal{C}$ be a class of graphs. If the graphs in $\mathcal{C}$ admit FP+C*-definable ordered treelike decompositions, then* FP+C *captures* PTIME *on $\mathcal{C}$.*

Capturing PTIME on Planar Graphs

Lemma

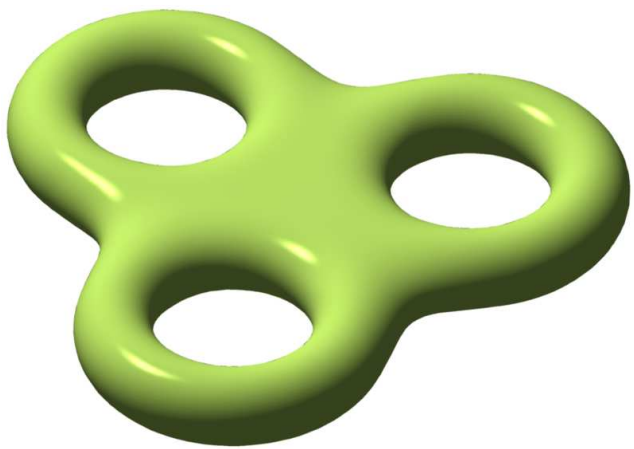
Let $\mathcal{C}$ be a class of graphs. If the graphs in $\mathcal{C}$ admit FP+C*-definable ordered treelike decompositions, then* FP+C *captures* PTIME *on $\mathcal{C}$.*

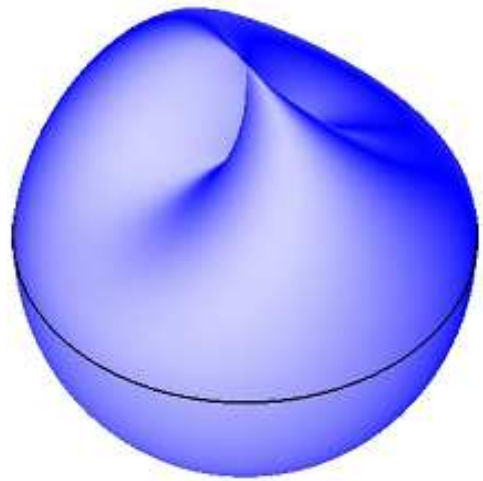
Corollary

FP+C *captures* PTIME *on the class of all planar graphs.*

Proof of the Main Theorem — Step 3:

Graphs of Embeddable in a Surface





Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits* FP*-definable ordered treelike decompositions.*

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits* FP*-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits FP-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits* FP*-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

Induction step:

For 3-connected graphs “polyhedrally” embedded into ***S***.

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits* FP*-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

Induction step:

For 3-connected graphs “polyhedrally” embedded into ***S***.

Try to find facial cycles as for 3-connected planar graphs. Two possible outcomes:

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits FP-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

Induction step:

For 3-connected graphs “polyhedrally” embedded into ***S***.

Try to find facial cycles as for 3-connected planar graphs. Two possible outcomes:

- All facial cycles are found. *Define order.*

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits* FP*-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

Induction step:

For 3-connected graphs “polyhedrally” embedded into ***S***.

Try to find facial cycles as for 3-connected planar graphs. Two possible outcomes:

- All facial cycles are found. *Define order.*
- Noncontractible cycle is found. *Delete it, apply induction hypothesis, lift decomposition to original graph.*

□

Extension to Surface Graphs

Lemma

*For every surface **S**, the class of all graphs embeddable into **S** admits FP-definable ordered treelike decompositions.*

Proof sketch.

Induction on the genus.

Base step: Planar Graphs.

Induction step:

For 3-connected graphs “polyhedrally” embedded into ***S***.

Try to find facial cycles as for 3-connected planar graphs. Two possible outcomes:

- All facial cycles are found. *Define order.*
- Noncontractible cycle is found. *Delete it, apply induction hypothesis, lift decomposition to original graph.*

Generalize to arbitrary graphs embeddable into ***S***. □

Extension to Surface Graphs

Lemma

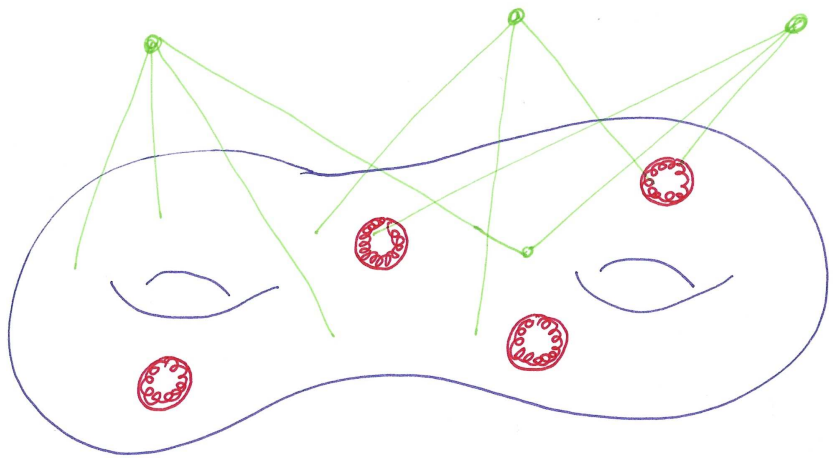
*For every surface **S**, the class of all graphs embeddable into **S** admits FP-definable ordered treelike decompositions.*

Corollary

*For every surface **S**, the logic FP+C capture PTIME on the class of graphs embeddable into **S**.*

Proof of the Main Theorem — Step 4:

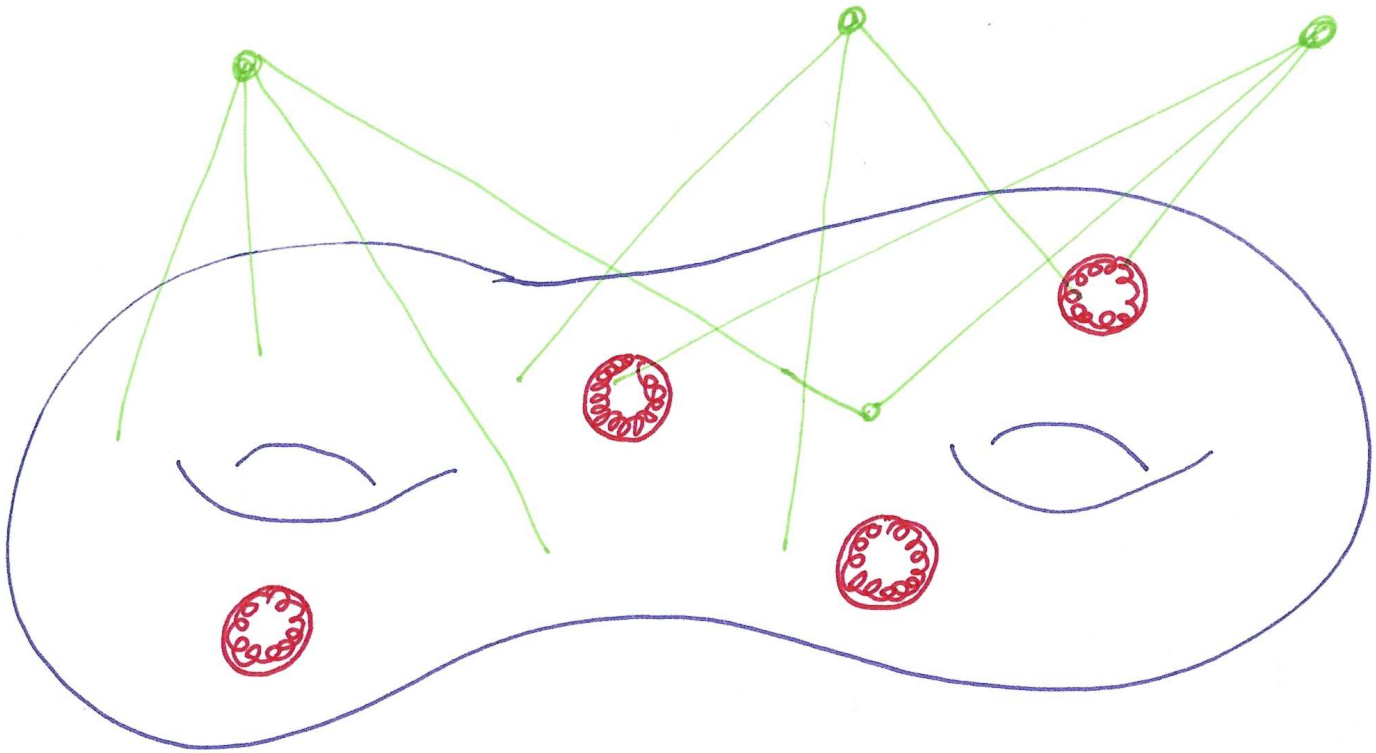
Graphs with Excluded Minors



Almost Embeddable Graph

G almost embeddable into ***S*** if *G* is embeddable into ***S*** up to:

- a bounded number of apices, vertices that have to be removed before the embedding
- a bounded number of vortices, highly nonplanar areas which nevertheless have a regular structure



Robertson and Seymour's Structure Theorem

Theorem (Robertson and Seymour 1984-1999)

*Let $\mathcal{C}$ be a class of graphs with an excluded minor. Then all graphs in $\mathcal{C}$ have tree decompositions into torsi that are almost embeddable into some surface **S**.*

The surface and the bounds on the number of vortices and apices depend on $\mathcal{C}$.

The Main Lemma

Lemma

Every class of graphs with excluded minors admits FP-definable ordered treelike decompositions.

The Main Lemma

Lemma

Every class of graphs with excluded minors admits FP-definable ordered treelike decompositions.

The Main Theorem follows.

Application to Isomorphism Testing

The *k*-dimensional Weisfeiler-Leman Algorithm (*k*-WL) is a simple combinatorial algorithm for testing isomorphism of graphs, based on an iterative “color refinement” on *k*-tuples of vertices.

Application to Isomorphism Testing

The k-dimensional Weisfeiler-Leman Algorithm (k-WL) is a simple combinatorial algorithm for testing isomorphism of graphs, based on an iterative "color refinement" on k-tuples of vertices.

k-WL has 1-sided error:

- If $G \cong H$, then k-WL will say so.
- If $G \ncong H$, then k-WL may fail to recongnize this.

Application to Isomorphism Testing

The k-dimensional Weisfeiler-Leman Algorithm (k-WL) is a simple combinatorial algorithm for testing isomorphism of graphs, based on an iterative "color refinement" on k-tuples of vertices.

k-WL has 1-sided error:

- If $G \cong H$, then k-WL will say so.
- If $G \ncong H$, then k-WL may fail to recongnize this.

Running Time is $n^{O(k)}$

Application to Isomorphism Testing

The k-dimensional Weisfeiler-Leman Algorithm (k-WL) is a simple combinatorial algorithm for testing isomorphism of graphs, based on an iterative "color refinement" on k-tuples of vertices.

k-WL has 1-sided error:

- If $G \cong H$, then k-WL will say so.
- If $G \ncong H$, then k-WL may fail to recongnize this.

Running Time is $n^{O(k)}$

Theorem

For every class $\mathcal{C}$ with excluded minors there is a k such that k-WL correctly decides isomorphism of graphs from $\mathcal{C}$.

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

FP+C cannot do linear algebra

Remember:

CFI-Theorem

FP+C *does not capture* PTIME*.*

FP+C cannot do linear algebra

Remember:

CFI-Theorem
FP+C *does not capture* PTIME.

The proof of the CFI-Theorem exploits the inability of FP+C to do basic linear algebra.
(Atserias, Bulatov, Dawar 2007, Dawar, Richerby, Rossman 2008)

Matrix Rank in FP+C

View formula $\phi(x, y)$ as defining a square $\{0, 1\}$-matrix.

Corollary (of the proof of the CFI-Theorem)

Matrix rank over GF(2) (or any other finite field) is not definable in FP+C*.*

Matrix Rank in FP+C

View formula $\phi(x, y)$ as defining a square $\{0, 1\}$-matrix.

Corollary (of the proof of the CFI-Theorem)

Matrix rank over GF(2) (or any other finite field) is not definable in FP+C*.*

Theorem (Dawar, G., Holm, Laubner)

Matrix rank over the rationals is definable in FP+C*.*

Fixed-Point Logic with Rank

FP + R = FP + matrix rank operator

Fixed-Point Logic with Rank

FP + R = FP + matrix rank operator

Theorem (Dawar, G., Holm, Laubner)

- FP+R *is strictly more expressive than* FP+C*.*
  In particular, all known examples of properties not definable in FP+C *are definable in* FP+R*.*

Fixed-Point Logic with Rank

FP + R = FP + matrix rank operator

Theorem (Dawar, G., Holm, Laubner)

- FP+R *is strictly more expressive than* FP+C*.*
  In particular, all known examples of properties not definable in FP+C *are definable in* FP+R*.*
- *Rank operators of increasing arity form strict hierarchy of expressiveness.*

Fixed-Point Logic with Rank

FP + R = FP + matrix rank operator

Theorem (Dawar, G., Holm, Laubner)

- FP+R *is strictly more expressive than* FP+C*.*
  In particular, all known examples of properties not definable in FP+C *are definable in* FP+R*.*
- *Rank operators of increasing arity form strict hierarchy of expressiveness.*

Many interesting open questions.

1. The Quest for a Logic Capturing PTIME
2. Fixed-Point Logic with Counting
3. Excluded Minors
4. Fixed-Point Logic with Rank
5. Open Problems

Further Classes of Graphs

Open Problem 1

Is there a natural logic that captures PTIME on classes of graphs of bounded degree ?

Further Classes of Graphs

Open Problem 1

Is there a natural logic that captures PTIME on classes of graphs of bounded degree ?

Open Problem 2

Is there a logic that captures PTIME on classes of graphs of bounded rank width ?

Open Problem 3

Does FP+R capture PTIME ?

Stronger Logics

Open Problem 3

Does FP+R capture PTIME ?

Choiceless polynomial time with counting CP+C, introduced by Blass, Gurevich and Shelah (1998), is another candidate for a logic capturing PTIME.

Open Problem 4

Does CP+C capture PTIME ?

Stronger Logics

Open Problem 3

Does FP+R capture PTIME ?

Choiceless polynomial time with counting CP+C, introduced by Blass, Gurevich and Shelah (1998), is another candidate for a logic capturing PTIME.

Open Problem 4

Does CP+C capture PTIME ?

Open Problem 5

What's the relation between CP+C and FP+R ?

Stronger Logics

Open Problem 3

Does FP+R capture PTIME ?

Choiceless polynomial time with counting CP+C, introduced by Blass, Gurevich and Shelah (1998), is another candidate for a logic capturing PTIME.

Open Problem 4

Does CP+C capture PTIME ?

Open Problem 5

What's the relation between CP+C and FP+R ?

Open Problems 6 and 7

Prove that FP+R and CP+C do not capture NP.

Smaller Complexity Classes

Open Problem 8

Is there a logic that captures uniform AC^0 ?

Smaller Complexity Classes

Open Problem 8

Is there a logic that captures uniform AC^0 ?

Open Problem 9

Is there an effective syntax for order invariant first-order logic ?