The combinatorics of monadic stability, monadic dependence, and related notions

Algomanet, Warsaw, September 9-13, 2024

Jan Dreier, TU Wien

Understanding **Monadic Stability** and **Monadic Dependence** via

- logic,
- combinatorics, and
- algorithms.

Map of the Universe

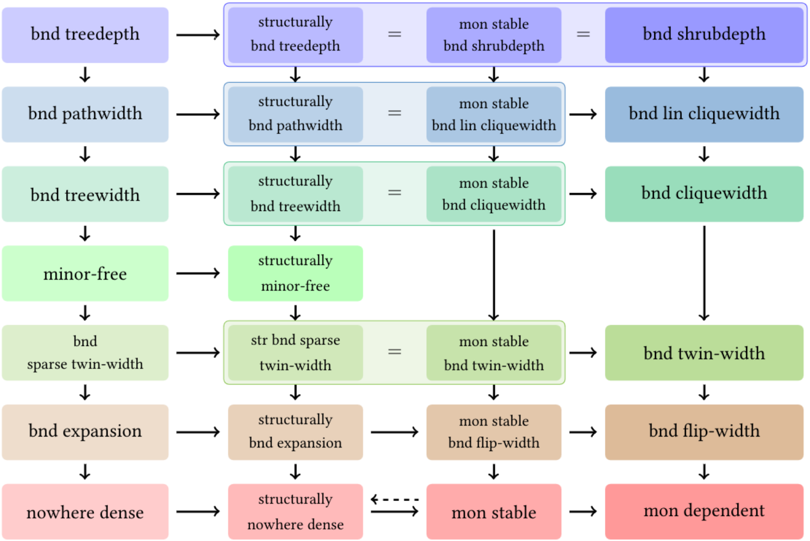


Figure by Michał Pilipczuk

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs
- Wednesday afternoon: exercises

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs
- Wednesday afternoon: exercises
- Thursday morning: neighborhood complexity and neighborhood covers

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs
- Wednesday afternoon: exercises
- Thursday morning: neighborhood complexity and neighborhood covers
- Thursday afternoon: exercises

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs
- Wednesday afternoon: exercises
- Thursday morning: neighborhood complexity and neighborhood covers
- Thursday afternoon: exercises
- Friday morning: pursuit-evasion games and flip-width

Schedule

- Monday morning: meta-theorems, logic, nowhere dense (+exercises)
- Monday afternoon: monadic stability and monadic dependence (+exercises)
- Tuesday morning: first-order model-checking
- Tuesday afternoon: exercises
- Wednesday morning: Ramsey and forbidden subgraphs
- Wednesday afternoon: exercises
- Thursday morning: neighborhood complexity and neighborhood covers
- Thursday afternoon: exercises
- Friday morning: pursuit-evasion games and flip-width
- Friday afternoon: exercises

Overview

Topics we will touch:

- parameterized complexity

Overview

Topics we will touch:

- parameterized complexity
- first-order logic

Overview

Topics we will touch:

- parameterized complexity
- first-order logic
- treewidth, treedepth, cliquewidth, etc.

Overview

Topics we will touch:

- parameterized complexity
- first-order logic
- treewidth, treedepth, cliquewidth, etc.
- nowhere dense classes

Overview

Topics we will touch:

- parameterized complexity
- first-order logic
- treewidth, treedepth, cliquewidth, etc.
- nowhere dense classes
- twin-width

Overview

Topics we will touch:

- parameterized complexity
- first-order logic
- treewidth, treedepth, cliquewidth, etc.
- nowhere dense classes
- twin-width
- monadic stability/dependence

Motivation

An Example

INDEPENDENTSET

- Input: Graph G and number k
- Question: Are there k pairwise non-adjacent vertices in G?

An Example

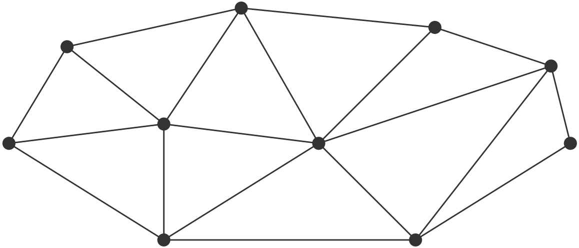
INDEPENDENTSET

- Input: Graph G and number k
- Question: Are there k pairwise non-adjacent vertices in G?

INDEPENDENTSET is NP-complete

Planar Graphs

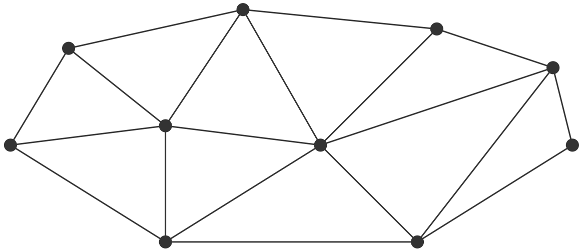
Does it help to restrict the input to certain "well-behaved" graphs?
For example planar graphs?



Planar Graphs

Does it help to restrict the input to certain “well-behaved” graphs?
For example planar graphs?

> INDEPENDENTSET is NP-complete on planar graphs.



Parameterized Complexity

Assign each instance a number, called the *parameter*. We hope that

- we can solve the instance if the parameter is small,
- interesting instances have a small parameter.

NP-hard problems may still be tractable for small parameter values!

Parameterized Independent Set

PARAMETERIZED INDEPENDENTSET

Input:	Graph G and integer k
Parameter:	k
Question:	Does G have an independent set of size k?

A parameterized problem is *fixed parameter tractable* (fpt) if instances with parameter k and size n can be solved in time $f(k)n^c$ (for some fixed function f and constant c).

Is PARAMETERIZED INDEPENDENTSET fixed parameter tractable?

Parameterized Hardness

The best known algorithm takes time $n^{\Theta(k)}$.

Parameterized Hardness

The best known algorithm takes time $n^{\Theta(k)}$.

for $v_1 \in V$
    for $v_2 \in V$
        ...
            for $v_k \in V$
                check if $v_1, \ldots, v_k$ is an IS of size k

This is optimal (under certain complexity assumptions).

Parameterized Hardness

The best known algorithm takes time $n^{\Theta(k)}$.

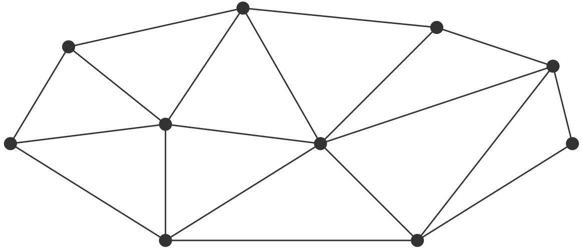
for $v_1 \in V$
  for $v_2 \in V$
    ...
      for $v_k \in V$
        check if $v_1, \ldots, v_k$ is an IS of size k

This is optimal (under certain complexity assumptions).

Parameterized IndependentSet is not fixed parameter tractable (unless FPT $=$ W[1]).

Independent Set on Planar Graphs

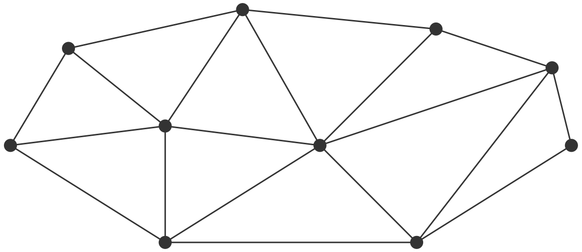
How about parameterized independent set on planar graphs?



Independent Set on Planar Graphs

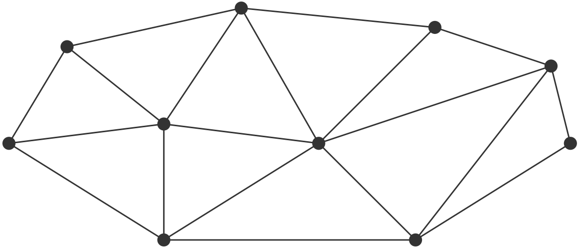
How about parameterized independent set on planar graphs?

PARAMETERIZED INDEPENDENTSET is fixed parameter tractable on planar graphs.



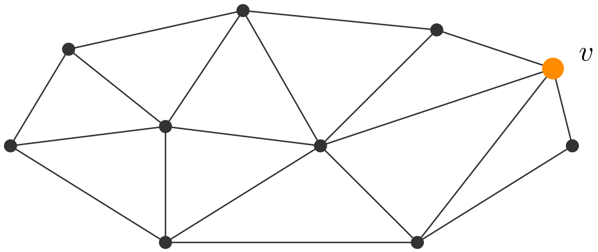
Independent Set on Planar Graphs

- We want to find an independent set of size k.



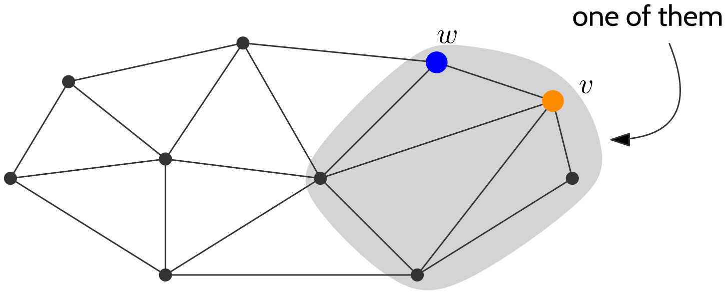
Independent Set on Planar Graphs

- We want to find an independent set of size k.
- In planar graphs there is always a vertex v with degree ≤ 5.



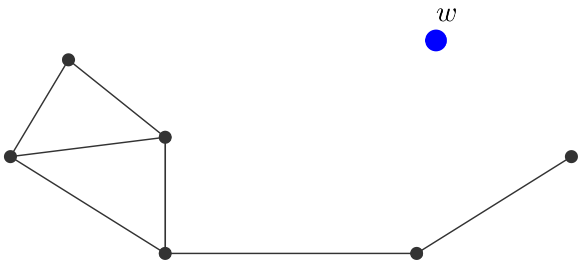
Independent Set on Planar Graphs

- We want to find an independent set of size k.
- In planar graphs there is always a vertex v with degree ≤ 5.
- At least one vertex w from $N(v)$ is in a maximal independent set.



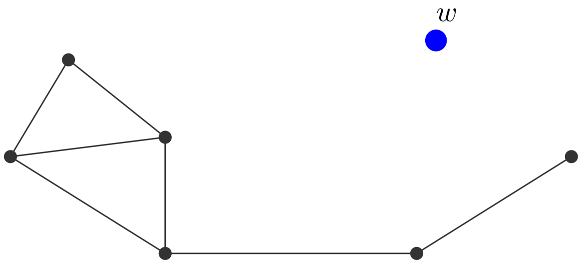
Independent Set on Planar Graphs

- We want to find an independent set of size k.
- In planar graphs there is always a vertex v with degree ≤ 5.
- At least one vertex w from $N(v)$ is in a maximal independent set.
- We guess w, place w in solution and remove $N(w)$.



Independent Set on Planar Graphs

- We want to find an independent set of size k.
- In planar graphs there is always a vertex v with degree ≤ 5.
- At least one vertex w from $N(v)$ is in a maximal independent set.
- We guess w, place w in solution and remove $N(w)$.
- Then find a solution of size $k-1$ in remaining graph.



Algorithm

IS(G, k):

  if G is empty return $k == 0$

  find vertex v with degree ≤ 5 in G

  for all $w \in N(v)$:

    if IS($G \setminus N(w)$, $k - 1$) return True

  return False

This solves PARAMETERIZED INDEPENDENTSET on planar graphs in time $O(6^k n)$.

Conclusion

INDEPENDENTSET is hard even if we

- consider only planar graphs, or
- parameterize by the solution size.

Conclusion

INDEPENDENTSET is hard even if we

- consider only planar graphs, or
- parameterize by the solution size.

But the problem becomes tractable if we both

- consider only planar graphs, and
- parameterize by the solution size.

Other Problems?

- Is parameterized Dominating Set FPT on planar graphs?
- Is parameterized Clique FPT on bounded genus graphs?
- ...

We would like a single mechanism that answers these and similar questions.

Algorithmic Meta-Theorems

Algorithmic Meta-Theorems:

"All Problems expressible in Logic L *can be solved efficiently on graph classes with property* P*"*

Algorithmic Meta-Theorems

Algorithmic Meta-Theorems:

"All Problems expressible in Logic L can be solved efficiently on graph classes with property P"

Our Goal:

- L is first-order logic
- P are monadically dependent graph classes

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure
- induced subgraph $\leftrightarrow$ substructure

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure
- induced subgraph $\leftrightarrow$ substructure

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure
- induced subgraph $\leftrightarrow$ substructure
- vertex $\leftrightarrow$ element
- all vertices of a graph $\leftrightarrow$ universe of the structure

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure
- induced subgraph $\leftrightarrow$ substructure
- vertex $\leftrightarrow$ element
- all vertices of a graph $\leftrightarrow$ universe of the structure
- adjacency $\leftrightarrow$ binary relation

Logic vs Graph-Theory

Graph-theorists and logicans use different languages:

- graph $\leftrightarrow$ structure
- induced subgraph $\leftrightarrow$ substructure
- vertex $\leftrightarrow$ element
- all vertices of a graph $\leftrightarrow$ universe of the structure
- adjacency $\leftrightarrow$ binary relation
- colors $\leftrightarrow$ unary relation

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

- Each structure has a *signature* τ: a set of relational symbols with given arities.

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

- Each structure has a *signature* τ: a set of relational symbols with given arities.
- We interpret *colored undirected graphs* as τ-structures with $\tau = \{E, c_1, c_2, \dots\}$ where

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

- Each structure has a *signature* τ: a set of relational symbols with given arities.
- We interpret *colored undirected graphs* as τ-structures with $\tau = \{E, c_1, c_2, \dots\}$ where
  - the universe are the vertices

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

- Each structure has a *signature* τ: a set of relational symbols with given arities.
- We interpret *colored undirected graphs* as τ-structures with $\tau = \{E, c_1, c_2, \dots\}$ where
  - the universe are the vertices
  - E denotes the binary adjacency relation between vertices

Graphs as Structures

Logic works on *structures*. We can easily see graphs as structures.

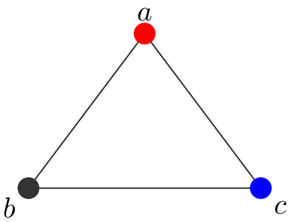
- Each structure has a *signature* τ: a set of relational symbols with given arities.

- We interpret *colored undirected graphs* as τ-structures with $\tau = \{E, c_1, c_2, \ldots\}$ where
  - the universe are the vertices
  - E denotes the binary adjacency relation between vertices
  - c_i denotes the unary relation “the vertex is colored with color i”

Example

This graph is a structure G with

- universe $V = \{a, b, c\}$
- symmetrical binary relation
  $E := \{(a, b), (b, a), (b, c), (c, b), (a, c), (c, a)\}$
- unary relations $c_1 := \{a\}$, $c_2 := \{c\}$



First-Order Logic

For a given signature τ, first-order logic has ...

- element-variables ($x, y, z, \dots$)
- the equality relation $=$ as well as the relations from τ.
- quantifiers $\exists$ and $\forall$, as well as operators $\wedge$, $\vee$ and $\neg$

We mostly work on colored undirected graphs with $\tau = \{E, c_1, c_2, \dots\}$.

Expressiveness

Can these properties be expressed in FO logic?

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \dots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \dots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

- The number of vertices is even.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

- The number of vertices is even. No.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

- The number of vertices is even. No.
- The graph is connected.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \dots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \dots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

- The number of vertices is even. No.
- The graph is connected. No.

Central Problem

First-Order Model-Checking

Input:	Graph G and first-order sentence φ
Question:	$G \models \varphi$?

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

FO model-checking on planar graphs in NP-hard.

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

FO model-checking on planar graphs in NP-hard.

Proof: Reduction from Independent Set.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

FO model-checking on planar graphs in NP-hard.

Proof: Reduction from Independent Set.

$$\exists x_1 \dots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

It is reasonable to assume that the length of the formula is small compared to the size of the graph. Parameterize by $|\varphi|$.

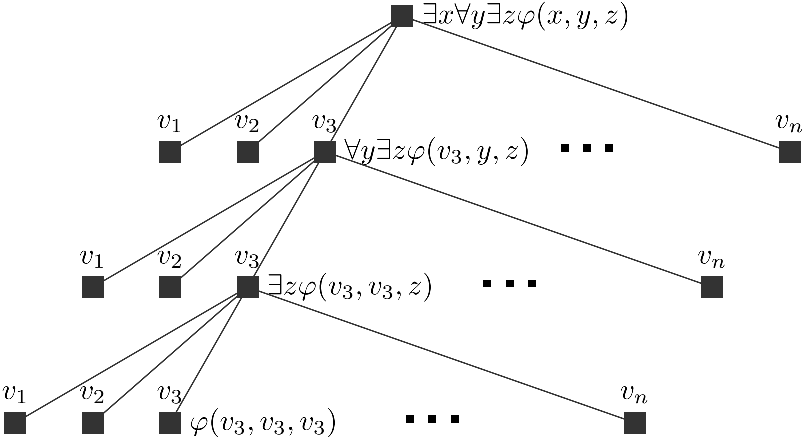
Parameterized Complexity (Upper Bound)

Theorem

One can decide whether $G \models \varphi$ in time $O(|G|^{|\varphi|})$.

Evaluation Trees

Proof: Construct an evaluation tree of size $O(|G|^{|\varphi|})$.



Parameterized Complexity (Lower Bound)

Conjecture (based on SETH)

One cannot decide whether $G \models \varphi$ in time $O(|G|^{q-1-\varepsilon})$ for any $\varepsilon > 0$ where q is the number of quantifiers of φ.

The previous algorithm is probably more or less optimal.

A faster model-checking algorithm would lead to an unexpected faster algorithm for many hard problems.

On certain graph classes, we can do much better though.

First-Order Logic

Target Statement

Let $\mathcal{C}$ be a "well-behaved" graph class. For an FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n^{10}$ for some function f.

First-Order Logic

Target Statement

Let $\mathcal{C}$ be a "well-behaved" graph class. For an FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n^{10}$ for some function f.

Examples of "well-behaved" classes:

- bounded degree
- planar graphs
- ...

Expressiveness

Every problem expressible in first-order logic …

Expressiveness

Every problem expressible in first-order logic ...

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

Expressiveness

Every problem expressible in first-order logic …

- There exists an independent set of size k.

$$\exists x_1 \dots \exists x_k \bigwedge_{i \neq j} \neg E(x_i, x_j) \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \dots \exists x_k \, \forall y \bigvee_{i} E(y, x_i) \vee y = x_i$$

…can be solved in time $f(k) \cdot n^{10}$ on well-behaved graph classes.

Exercise

Does the algorithic meta-theorem give an fpt algorithm for the following problem?

- Input: SAT-instance with planar incidence graph, $k \in \mathbb{N}$.
- Parameter: k.
- Question: is there a satisfying assignment with at most k variables set to true?

Exercise

Does the algorithic meta-theorem give an fpt algorithm for the following problem?

- Input: a planar graph G, and $k \in \mathbb{N}$.
- Parameter: k.
- Is there a dominating set of size at most k that induces a connected subgraph?

Exercise

Does the algorithic meta-theorem give an fpt algorithm for the following problem?

- Input: a planar graph G, and $k, d \in \mathbb{N}$.
- Parameter: k.
- Is it possible to remove k vertices such that every vertex has degree at most d?

What other graph classes are “well-behaved”?

Many Sparse Graph Classes

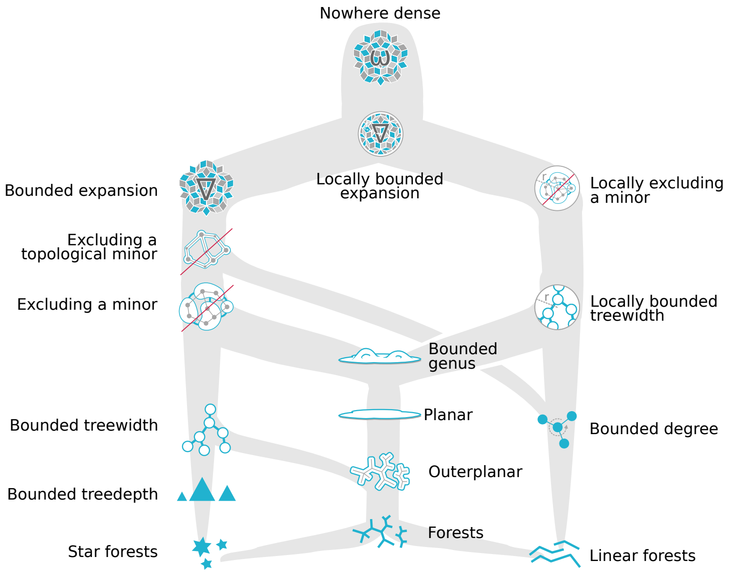


Figure by Felix Reidl

Milestones

Theorem (Grohe, Kreutzer, Siebertz 2017)

For a graph class $\mathcal{C}$ that is closed under subgraphs holds:
$\mathcal{C}$ is nowhere dense iff the first-order model-checking problem on $\mathcal{C}$ is FPT (assuming FPT $\neq$ AW[$*$]).

Bounded Degree Model Checking: Seese, 1996
Planar Model Checking: Flum, Grohe 2001
Bounded Expansion Model Checking: Dvořák, Král, Thomas, 2010
Nowhere Dense Model Checking: Grohe, Kreutzer, Siebertz, 2017

Nowhere Dense

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that no graph in $\mathcal{C}$ contains the r-subdivided clique of size k as a subgraph.

Nowhere Dense

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that no graph in $\mathcal{C}$ contains the r-subdivided clique of size k as a subgraph.

Let $\mathcal{C}$ be nowhere dense. Prove that there exists t such that no graph in $\mathcal{C}$ contains the biclique $K_{t,t}$ as a subgraph.

Nowhere Dense

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that no graph in $\mathcal{C}$ contains the r-subdivided clique of size k as a subgraph.

Let $\mathcal{C}$ be nowhere dense. Prove that there exists t such that no graph in $\mathcal{C}$ contains the biclique $K_{t,t}$ as a subgraph.

Prove that the class of half-graphs is not nowhere dense.

Nowhere Dense

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that no graph in $\mathcal{C}$ contains the r-subdivided clique of size k as a subgraph.

Let $\mathcal{C}$ be nowhere dense. Prove that there exists t such that no graph in $\mathcal{C}$ contains the biclique $K_{t,t}$ as a subgraph.

Prove that the class of half-graphs is not nowhere dense.

Prove that the class of trees is nowhere dense

Nowhere Dense

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that no graph in $\mathcal{C}$ contains the r-subdivided clique of size k as a subgraph.

Let $\mathcal{C}$ be nowhere dense. Prove that there exists t such that no graph in $\mathcal{C}$ contains the biclique $K_{t,t}$ as a subgraph.

Prove that the class of half-graphs is not nowhere dense.

Prove that the class of trees is nowhere dense

Prove that every class of bounded degree is nowhere dense.

Many Sparse Graph Classes

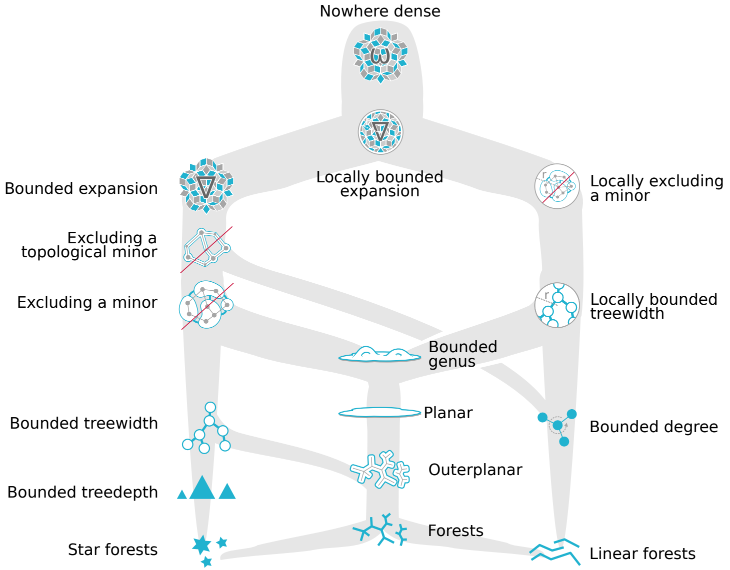
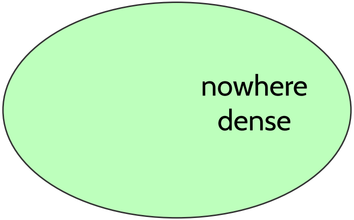


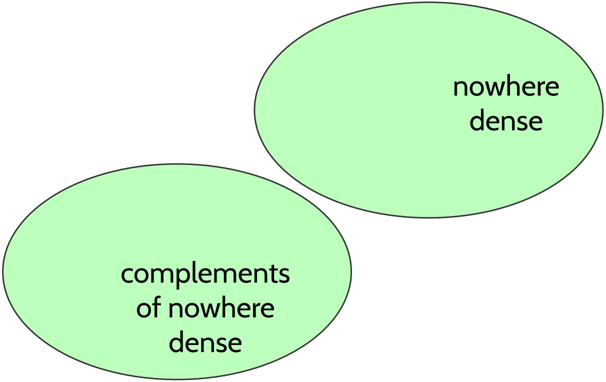
Figure by Felix Reidl

Can we go beyond nowhere dense?

Classes with FPT first-order model-checking?

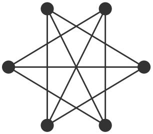


Classes with FPT first-order model-checking?



Complements

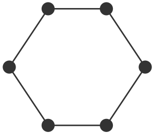
First-order model-checking is fpt on complements of nowhere dense classes by reduction.



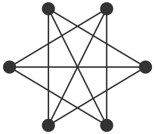
$\bar{G} \models \varphi$

Complements

First-order model-checking is fpt on complements of nowhere dense classes by reduction.



$$\Leftrightarrow$$

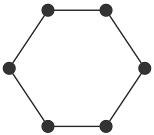


$G \models \bar{\varphi}$

$\bar{G} \models \varphi$

Complements

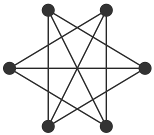
First-order model-checking is fpt on complements of nowhere dense classes by reduction.



$G \models \bar{\varphi}$

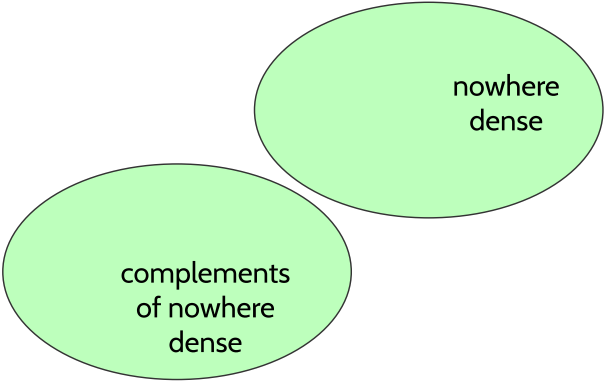
obtain $\bar{\varphi}$ from φ by swapping $\sim$ and $\not\sim$

$\Leftrightarrow$

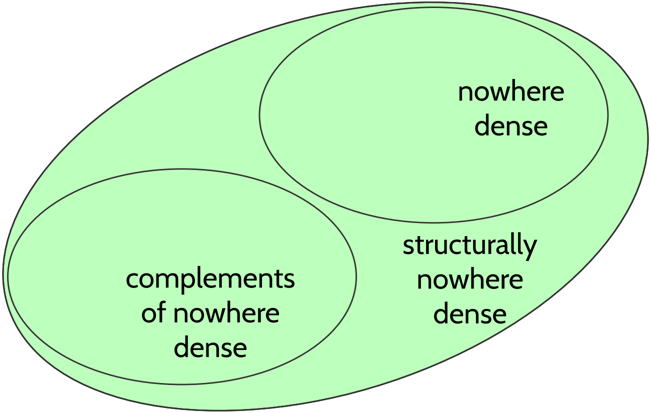


$\bar{G} \models \varphi$

Classes with FPT first-order model-checking?

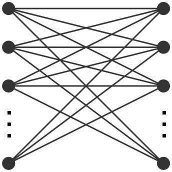


Classes with FPT first-order model-checking?



Fully Bipartite Graphs

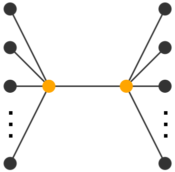
Can we do fpt model-checking on the class of fully bipartite graphs?



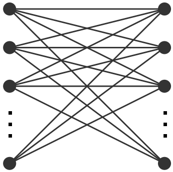
$G \models \varphi$

Fully Bipartite Graphs

Can we do fpt model-checking on the class of fully bipartite graphs?

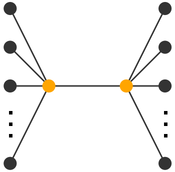


$G' \models \varphi'$



$G \models \varphi$

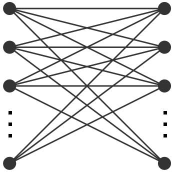
Can we do fpt model-checking on the class of fully bipartite graphs?



$G' \models \varphi'$

obtain φ' from φ by replacing $x{\sim}y$ with dist$(x, y) = 3$

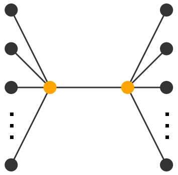
$\Leftrightarrow$



$G \models \varphi$

Fully Bipartite Graphs

Can we do fpt model-checking on the class of fully bipartite graphs?

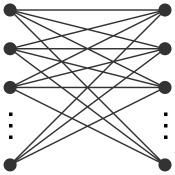


$G' \models \varphi'$

obtain φ' from φ by replacing $x{\sim}y$ with $\mathrm{dist}(x,y) = 3$

$\Longleftrightarrow$

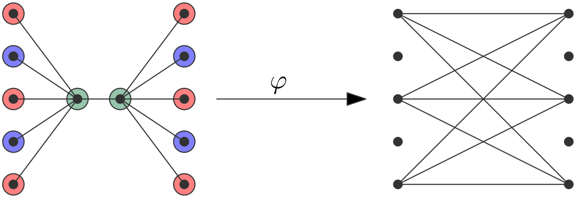
also restrict quantifiers to black vertices



$G \models \varphi$

Transductions (see board)

φ-transduction: color vertices + apply φ + take induced subgraph

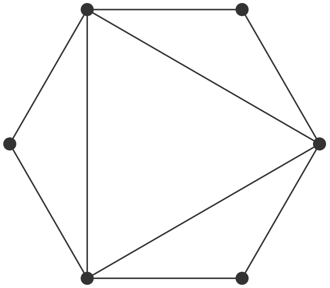


$\varphi(x, y) :=$ Red$(x) \wedge$ Red$(y) \wedge$ dist$(x, y) = 3$

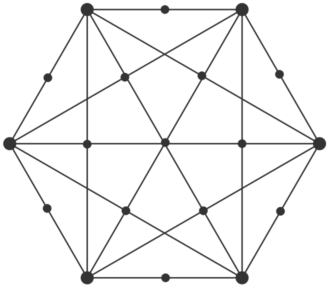
A class $\mathcal{D}$ is a *transduction* of a class $\mathcal{C}$ if there exists φ such that every graph in $\mathcal{D}$ is a φ-transduction of some graph in $\mathcal{C}$.

Transductions

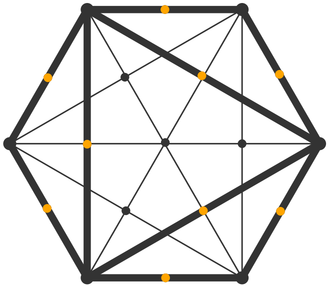
The class of subdivided cliques transduces the class of all graphs.



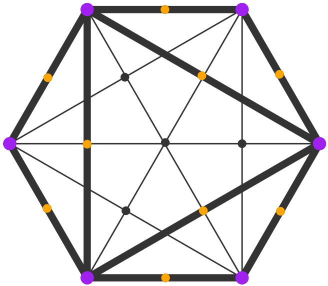
The class of subdivided cliques transduces the class of all graphs.



The class of subdivided cliques transduces the class of all graphs.

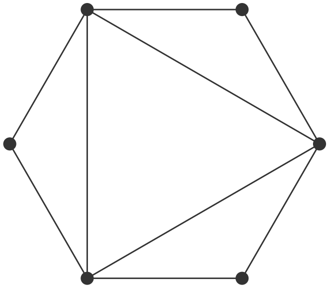


The class of subdivided cliques transduces the class of all graphs.



Transductions

The class of subdivided cliques transduces the class of all graphs.



Tractable Classes

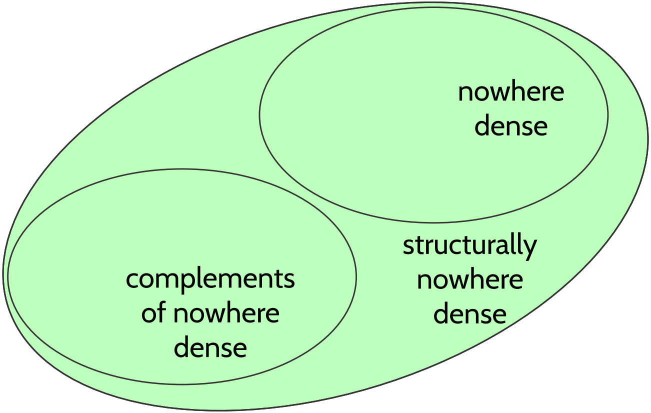
Gajarský, Kreutzer, Něsetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk, 2018.
Něsetřil, Ossona de Mendez, 2016

A class is *structurally nowhere dense*, if it is a transduction of a nowhere dense graph class.

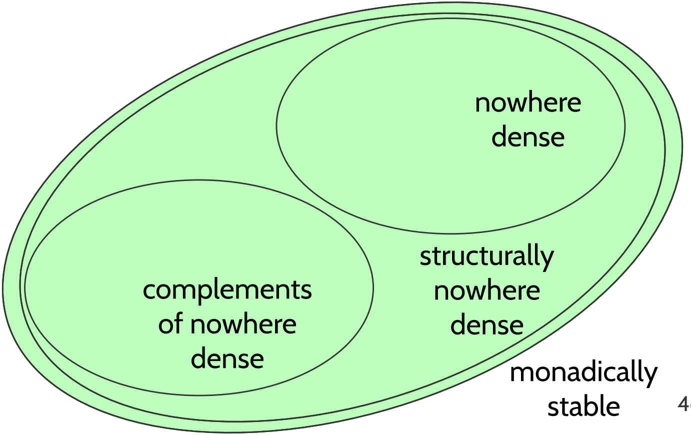
D, Mählmann, Siebertz, 2023

The first-order model-checking problem on $\mathcal{C}$ is FPT on structurally nowhere dense graph classes.

Classes with FPT first-order model-checking?



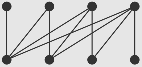
Classes with FPT first-order model-checking?



Monadic Stability/Dependence

Baldwin, Shelah, 1985

A class is *monadically stable*, if it does not transduce the class of all half-graphs.

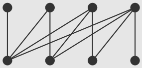


A class is *monadically dependent*, if it does not transduce the class of all graphs.

Monadic Stability/Dependence

Baldwin, Shelah, 1985

A class is *monadically stable*, if it does not transduce the class of all half-graphs.



A class is *monadically dependent*, if it does not transduce the class of all graphs.

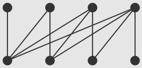
Adler, Adler

Every structurally nowhere dense class is monadically stable.

Monadic Stability/Dependence

Baldwin, Shelah, 1985

A class is *monadically stable*, if it does not transduce the class of all half-graphs.



A class is *monadically dependent*, if it does not transduce the class of all graphs.

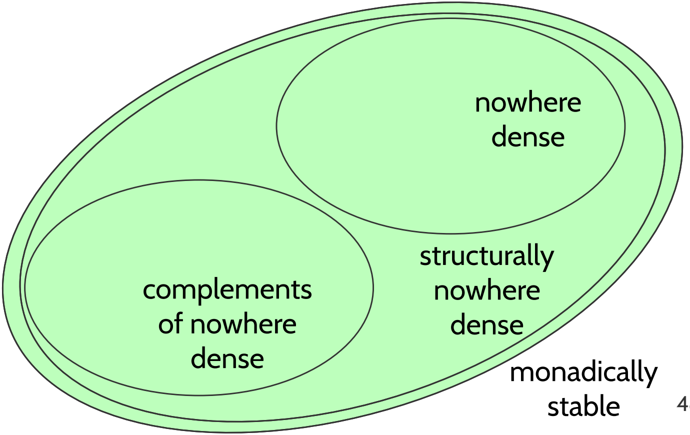
Adler, Adler

Every structurally nowhere dense class is monadically stable.

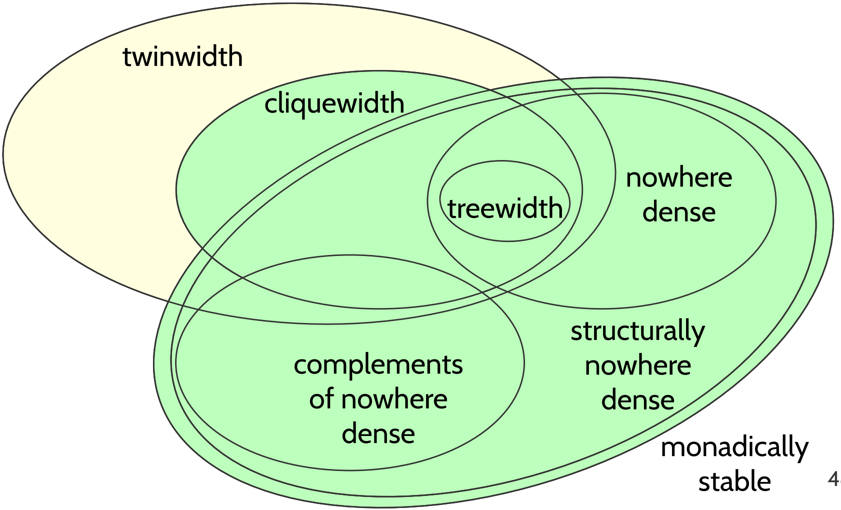
D, Eleftheriadis, Mählmann, McCarty, Pilipczuk, Toruńczyk 2024

Let $\mathcal{C}$ be monadically stable. The first-order model-checking problem is FPT on $\mathcal{C}$.

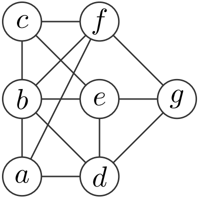
Classes with FPT first-order model-checking?



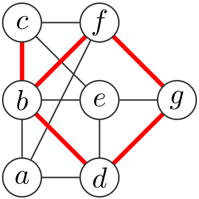
Classes with FPT first-order model-checking?



You already know normal graphs.

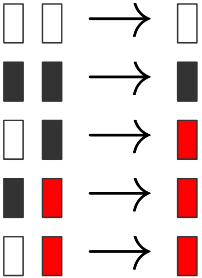


You already know normal graphs.

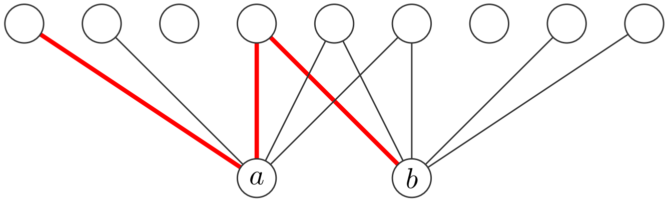


In *trigraphs* there are additional red error edges.

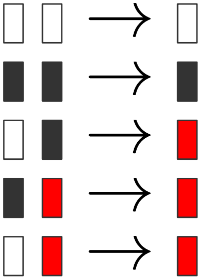
Contraction

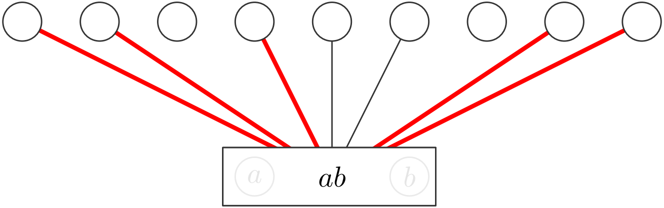


We can contract two (not neccessarily adjacent) vertices a and b. The edges of the new vertex ab follow this table.

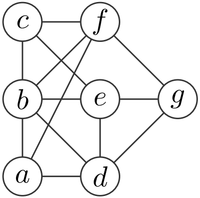


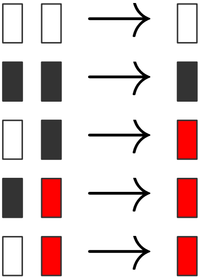
We can contract two (not neccessarily adjacent) vertices a and b. The edges of the new vertex ab follow this table.





Contraction Sequences

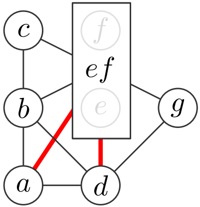


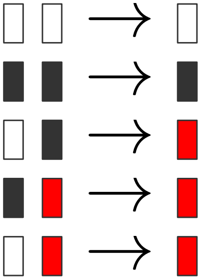


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

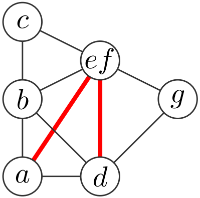


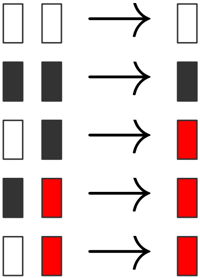


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

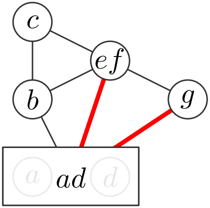


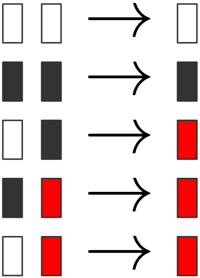


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

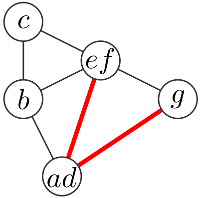


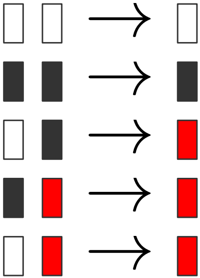


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

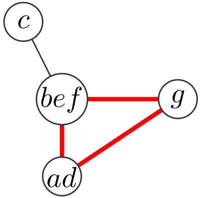


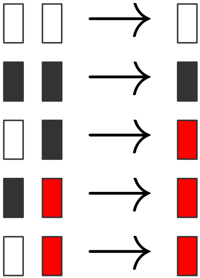


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

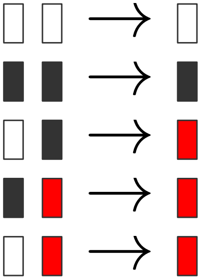


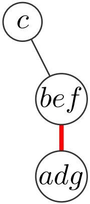


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

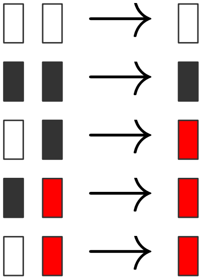


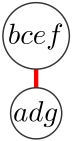


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

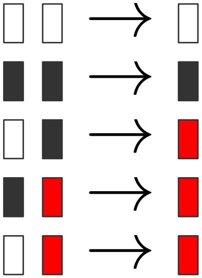




A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

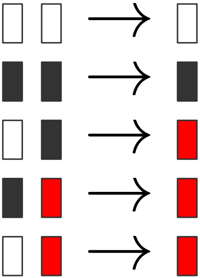
Contraction Sequences

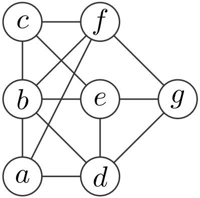


A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet

Contraction Sequences

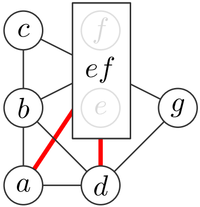


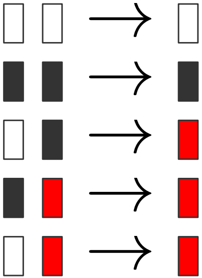


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

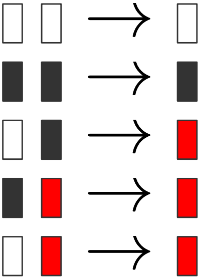


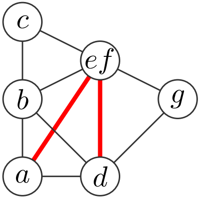


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

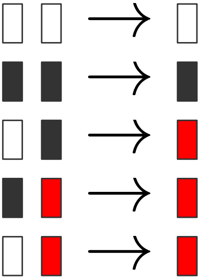


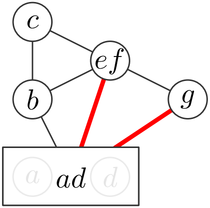


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

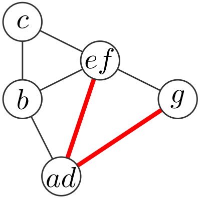


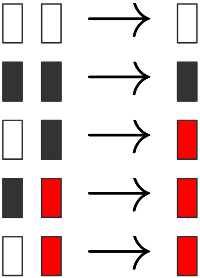


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

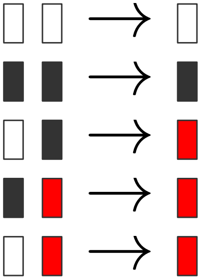


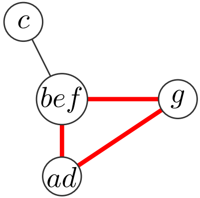


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

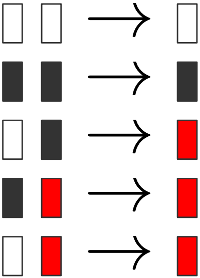


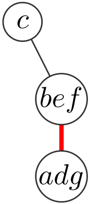


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

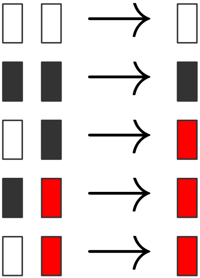


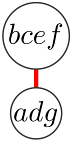


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences

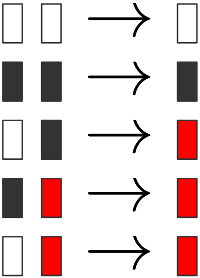


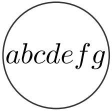


Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Contraction Sequences





Bonnet, Kim, Thomassé, Watrigant 2021

Twinwidth: Smallest integer d such there is a contraction sequence where the red degree is *at all times* at most d.

Classes of Bounded Twinwidth

The following classes have bounded twinwidth

- planar graphs,
- classes with bounded cliquewidth.

Classes of Bounded Twinwidth

The following classes have bounded twinwidth

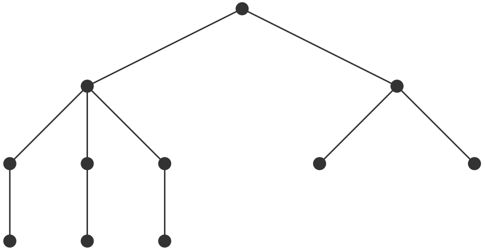
- planar graphs,
- classes with bounded cliquewidth.

The following classes do not have bounded twinwidth

- graphs with degree three.

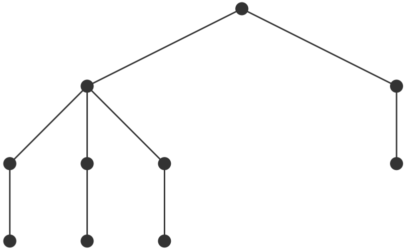
Twinwidth of Trees

Trees have twinwidth at most two. Strategy:

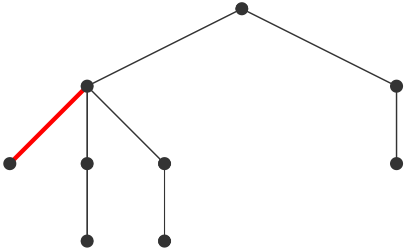


Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs.

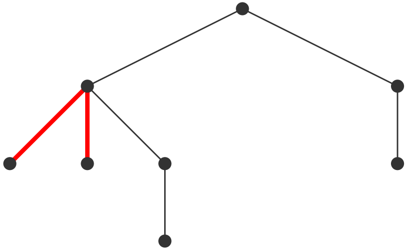


Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

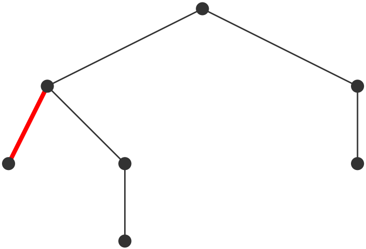


Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

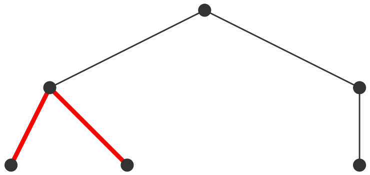


Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

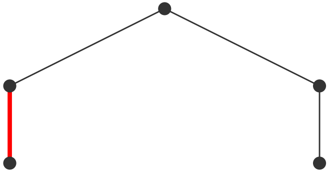


Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

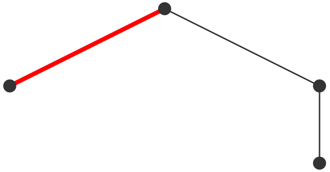


Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.



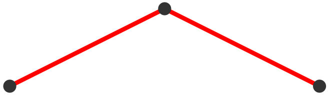
Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.



Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.



Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

Twinwidth of Trees

Trees have twinwidth at most two. Strategy: When possible contract twin leafs. Otherwise contract a deepest leaf with its parent.

First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth.

First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth. But once we do, model-checking is fpt.

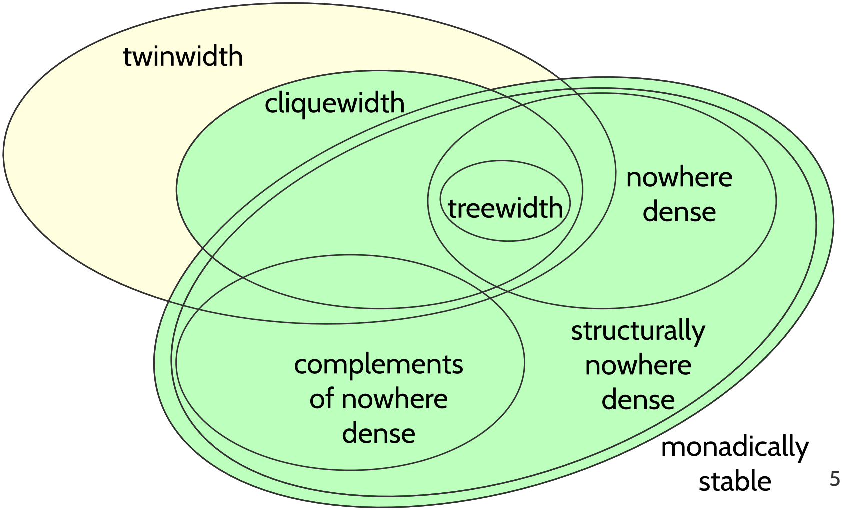
First-Order Model-Checking

So far, nobody knows how to compute (approximate) contraction sequences of graphs with bounded twinwidth. But once we do, model-checking is fpt.

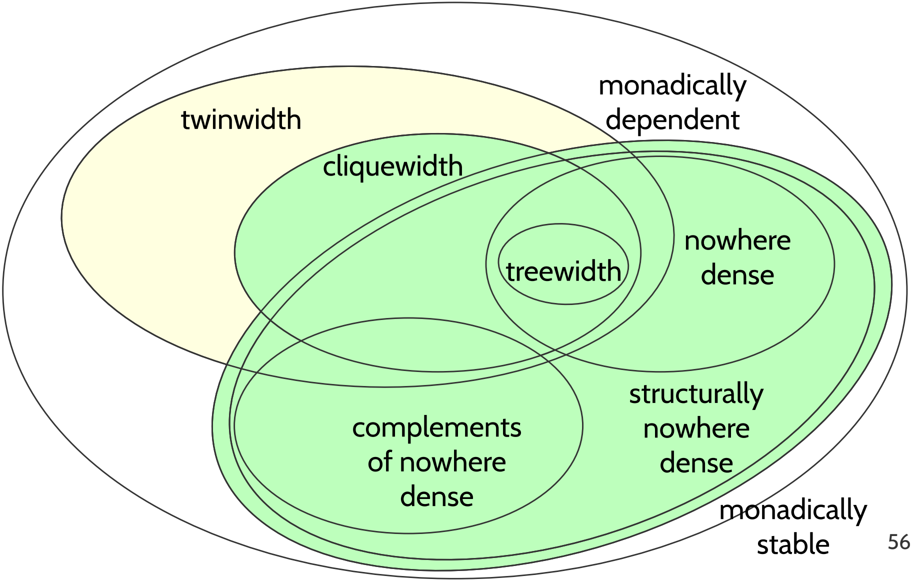
Bonnet, Kim, Thomassé, Watrigant 2021

Let $\mathcal{C}$ be a class of bounded twinwidth. Then first-order model-checking is fpt on $\mathcal{C}$, if one is additionally provided a contraction sequence of bounded twinwidth.

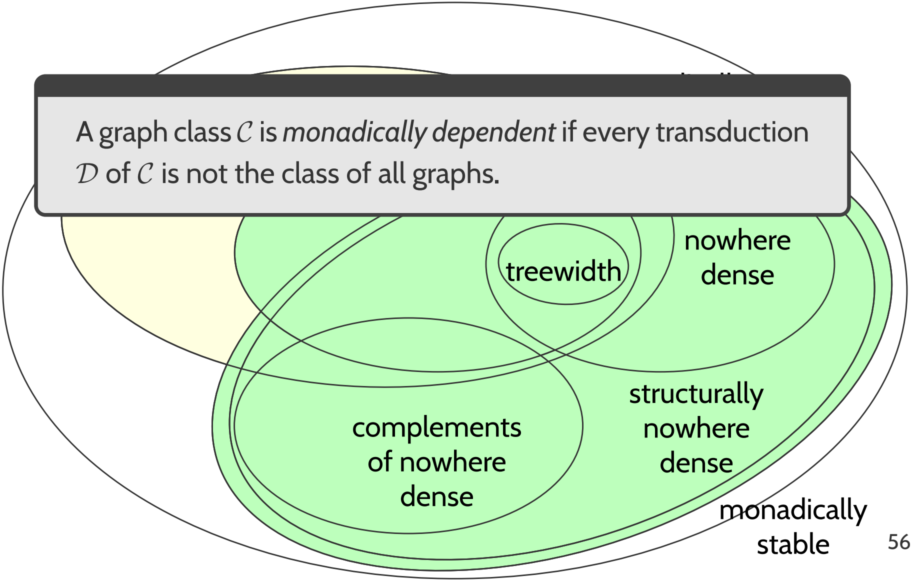
Classes with FPT first-order model-checking?



Classes with FPT first-order model-checking?



Classes with FPT first-order model-checking?



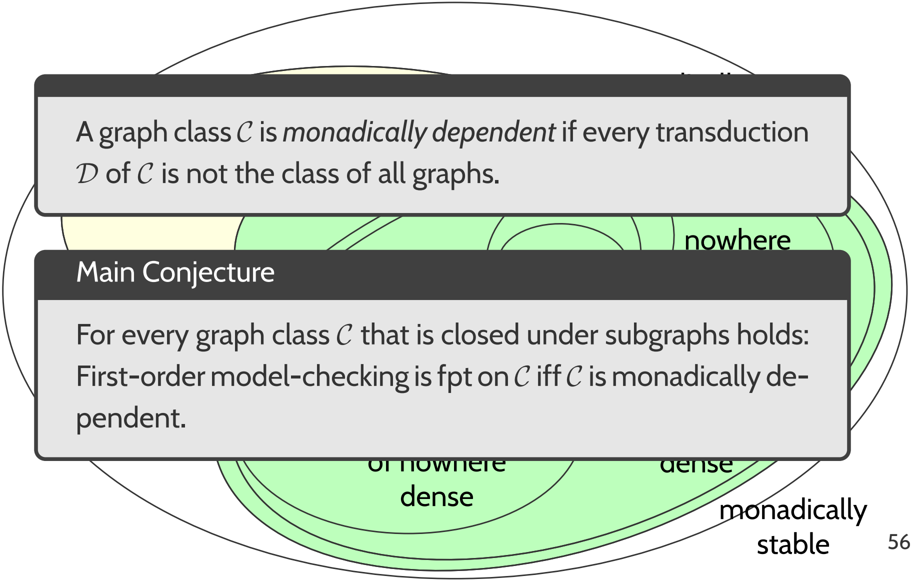
A graph class $\mathcal{C}$ is *monadically dependent* if every transduction $\mathcal{D}$ of $\mathcal{C}$ is not the class of all graphs.

Classes with FPT first-order model-checking?

A graph class $\mathcal{C}$ is *monadically dependent* if every transduction $\mathcal{D}$ of $\mathcal{C}$ is not the class of all graphs.

Main Conjecture

For every graph class $\mathcal{C}$ that is closed under subgraphs holds: First-order model-checking is fpt on $\mathcal{C}$ iff $\mathcal{C}$ is monadically dependent.



Map of the Universe

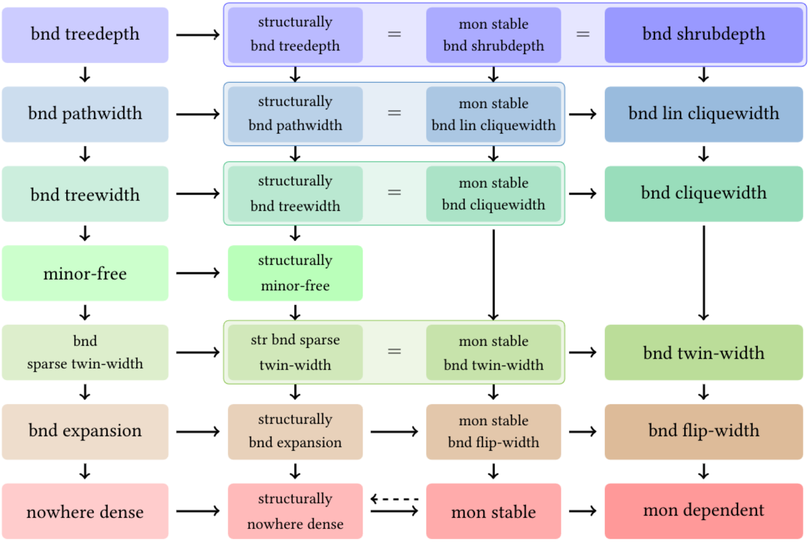


Figure by Michał Pilipczuk

Exercises

Exercise

Show that the class of empty graphs is monadically stable/dependent.

Exercise

Argue: If $\mathcal{C}$ transduces $\mathcal{D}$ and $\mathcal{D}$ transduces $\mathcal{E}$, then $\mathcal{C}$ transduces $\mathcal{E}$.

Exercise

Argue: If $\mathcal{C}$ transduces $\mathcal{D}$ and $\mathcal{D}$ transduces $\mathcal{E}$, then $\mathcal{C}$ transduces $\mathcal{E}$.

Argue that therefore monadically stable/dependent classes are closed under transductions.

Exercise

Let $\mathcal{C}$ be the class of graphs of degree at most three. Show that $\mathcal{C}$ is monadically stable/dependent.

Exercise

Let $\mathcal{C}$ be the class of graphs of degree at most three. Show that $\mathcal{C}$ is monadically stable/dependent.

Use the following theorem:

Corollary of Gaifman's Theorem

Let φ be a first-order formula. There is a number k with the following property. For every graph G there is a coloring $c : V(G) \to [k]$ such that for all $u, v \in V(G)$ with distance larger than k, the fact whether $(u, v) \in E(G)$ depends only on $(c(u), c(v))$.

Exercise

Show that the class of “star matchings” and the class of “comparability grids” are not monadically dependent.

Appendix

Restating the Milestones

For every graph class $\mathcal{C}$ that is closed under subgraphs:

$\mathcal{C}$ is nowhere dense if and only if $\mathcal{C}$ is monadically dependent.

Restating the Milestones

For every graph class $\mathcal{C}$ that is closed under subgraphs:

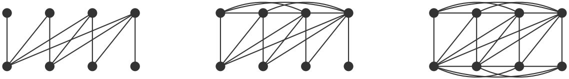
$\mathcal{C}$ is nowhere dense if and only if $\mathcal{C}$ is monadically dependent.

Grohe, Kreutzer, Siebertz 2017

For a graph class $\mathcal{C}$ that is closed under subgraphs holds:
$\mathcal{C}$ is monadically dependent iff the first-order model-checking problem on $\mathcal{C}$ is FPT (assuming FPT $\neq$ AW[$*$]).

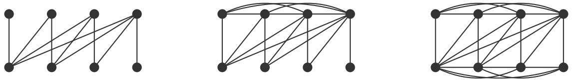
Restating the Milestones

A graph class $\mathcal{C}$ is *unordered* if for some k it excludes the following graphs of order k as induced subgraphs.



Restating the Milestones

A graph class $\mathcal{C}$ is *unordered* if for some k it excludes the following graphs of order k as induced subgraphs.

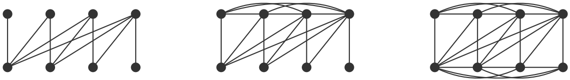


For every unordered graph class $\mathcal{C}$:

$\mathcal{C}$ is monadically stable if and only if $\mathcal{C}$ is monadically dependent.

Restating the Milestones

A graph class $\mathcal{C}$ is *unordered* if for some k it excludes the following graphs of order k as induced subgraphs.



For every unordered graph class $\mathcal{C}$:

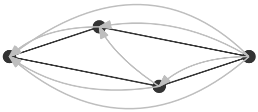
$\mathcal{C}$ is monadically stable if and only if $\mathcal{C}$ is monadically dependent.

D, Eleftheriadis, Mählmann, McCarty, Pilipczuk, Toruńczyk 2024

For an unordered graph class $\mathcal{C}$ that is closed under induced subgraphs holds:
$\mathcal{C}$ is monadically dependent iff the first-order model-checking problem on $\mathcal{C}$ is FPT (assuming FPT $\neq$ AW[$*$]).

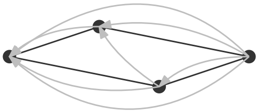
Restating the Milestones

An *ordered graph* is a graph together with a total order on its vertices (which can be queried by first-order logic).



Restating the Milestones

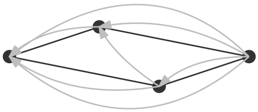
An *ordered graph* is a graph together with a total order on its vertices (which can be queried by first-order logic).



For every class of ordered graphs $\mathcal{C}$: $\mathcal{C}$ has bounded twin-width if and only if $\mathcal{C}$ is monadically dependent.

Restating the Milestones

An *ordered graph* is a graph together with a total order on its vertices (which can be queried by first-order logic).



For every class of ordered graphs $\mathcal{C}$: $\mathcal{C}$ has bounded twin-width if and only if $\mathcal{C}$ is monadically dependent.

Bonnet, Giocanti, Ossona de Mendez, Simon, Thomassé, Toruńczyk 2021

For a class $\mathcal{C}$ of ordered graphs that is closed under induced subgraphs holds: $\mathcal{C}$ is monadically dependent iff the first-order model-checking problem on $\mathcal{C}$ is FPT (assuming FPT $\neq$ AW[$*$]).

Map of the Universe

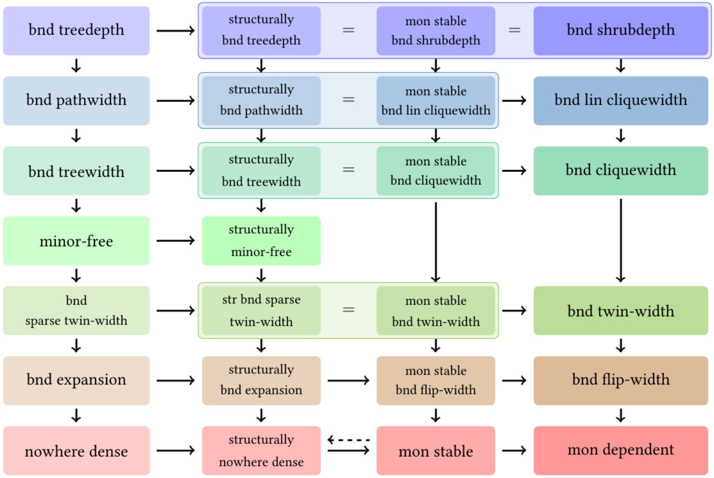


Figure by Michał Pilipczuk