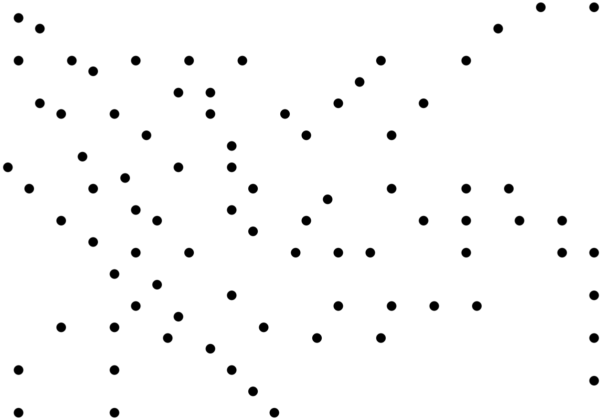
Monadically Stable and Monadically Dependent Graph Classes

Characterizations and Algorithmic Meta-Theorems

Nikolas Mählmann

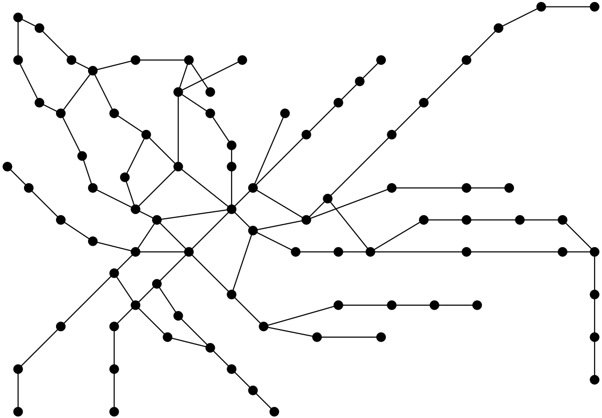
05.09.2024, PhD defense

Graphs



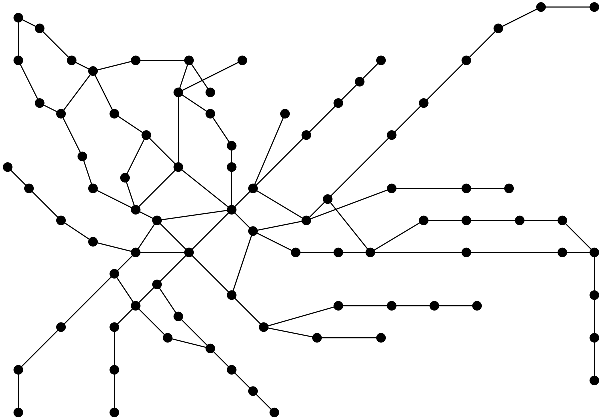
A *graph* consists of *vertices* connected by *edges*.

Graphs



A *graph* consists of *vertices* connected by *edges*.

Graphs



A *graph* consists of *vertices* connected by *edges*.

Graphs are an effective way to model real systems:

- road networks
- power grids
- computer networks
- circuits
- molecules

Graphs

A *graph* consists of *vertices* connected by *edges*.

Graphs are an effective way to model real systems:

- road networks
- power grids
- computer networks
- circuits
- molecules

Graphs

A *graph* consists of *vertices* connected by *edges*.

Graphs are an effective way to model real systems:

- road networks
- power grids
- computer networks
- circuits
- molecules

A *graph class* is a (usually infinite) set of graphs. Example: the class of all *cliques*:

The FO Model Checking Problem

Problem: Given a graph G and an FO sentence φ, decide whether

$$G \models \varphi.$$

Example: G contains a dominating set of size k iff.

$$G \models \exists x_1 \ldots \exists x_k \forall y : \bigvee_{i \in [k]} \big(y = x_i \vee \mathrm{Edge}(y, x_i)\big).$$

The FO Model Checking Problem

Problem: Given a graph G and an FO sentence φ, decide whether

$$G \models \varphi.$$

Example: G contains a dominating set of size k iff.

$$G \models \exists x_1 \dots \exists x_k \forall y : \bigvee_{i \in [k]} \big(y = x_i \vee \mathrm{Edge}(y, x_i)\big).$$

Further expressible problems: Independent Set, Subgraph Isomorphism, Independent Red-Blue Distance-7 Dominating Set, ...

The FO Model Checking Problem

Problem: Given a graph G and an FO sentence φ, decide whether

$$G \models \varphi.$$

Example: G contains a dominating set of size k iff.

$$G \models \exists x_1 \ldots \exists x_k \forall y : \bigvee_{i \in [k]} \big(y = x_i \vee \mathrm{Edge}(y, x_i)\big).$$

Further expressible problems: Independent Set, Subgraph Isomorphism, Independent Red-Blue Distance-7 Dominating Set, ...

Runtime: Let q be the quantifier rank of φ. On the class of all graphs, the naive $\mathcal{O}(n^q)$ algorithm is best possible, assuming ETH.

The FO Model Checking Problem

Problem: Given a graph G and an FO sentence φ, decide whether

$$G \models \varphi.$$

Example: G contains a dominating set of size k iff.

$$G \models \exists x_1 \ldots \exists x_k \forall y : \bigvee_{i \in [k]} \big(y = x_i \vee \mathrm{Edge}(y, x_i)\big).$$

Further expressible problems: Independent Set, Subgraph Isomorphism, Independent Red-Blue Distance-7 Dominating Set, ...

Runtime: Let q be the quantifier rank of φ. On the class of all graphs, the naive $\mathcal{O}(n^q)$ algorithm is best possible, assuming ETH.

Question: On which classes is FO model checking fixed-parameter tractable, i.e., solvable in time $f(\varphi) \cdot n^c$?

Nowhere Dense Classes of Graphs

For sparse graph classes, we know the exact limits of tractability.

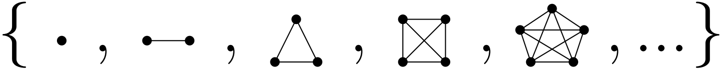
Theorem [Grohe, Kreutzer, Siebertz, 2014]

Let $\mathcal{C}$ be a *monotone* graph class.

- If $\mathcal{C}$ is *nowhere dense*, then model checking is fixed-parameter tractable on $\mathcal{C}$.
- Otherwise model checking is AW[$*$]-hard on $\mathcal{C}$.

Nowhere denseness generalizes many notions of sparsity such as: bounded degree, bounded tree-width, planarity, excluding a minor, ...

Monotone and Hereditary Graph Classes



The class of all cliques is not nowhere dense, but model checking is trivial there.

Monotone and Hereditary Graph Classes

The class of all cliques is not nowhere dense, but model checking is trivial there.

Cliques are not *monotone*: closed under taking subgraphs.
(i.e. deleting vertices and edges)

Monotone and Hereditary Graph Classes

The class of all cliques is not nowhere dense, but model checking is trivial there.

Cliques are not *monotone*: closed under taking subgraphs.
(i.e. deleting vertices and edges)

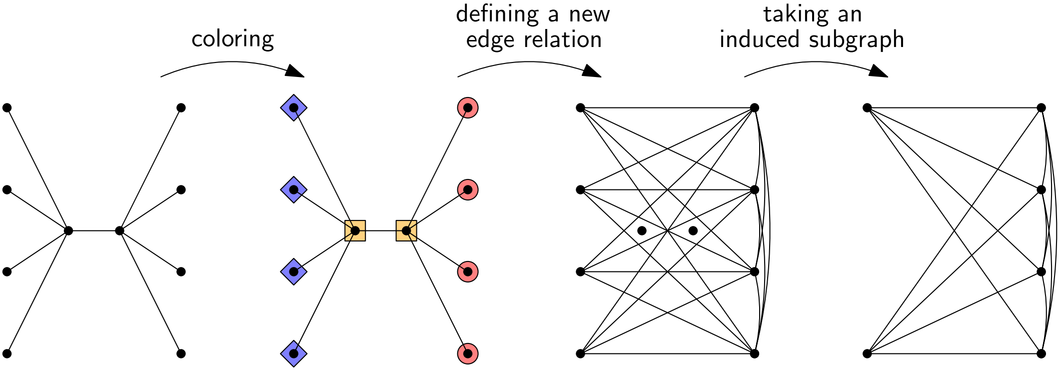
But cliques are *hereditary*: closed under taking induced subgraphs.
(i.e. deleting vertices)

To go beyond sparse classes, we need to shift from monotone to hereditary classes.

Transductions

Transductions are graph transformations defined by FO logic.

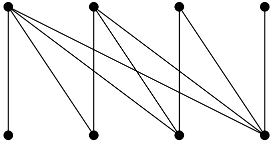
Example: $\varphi(x, y) = (\mathrm{dist}(x, y) = 3) \vee (\mathrm{Red}(x) \wedge \mathrm{Red}(y))$



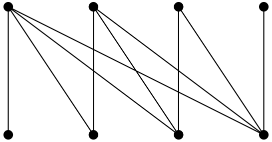
Monadic Stability and Monadic Dependence

Definition

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

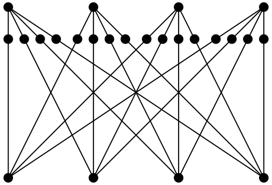


Monadic Stability and Monadic Dependence



Definition

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



Definition

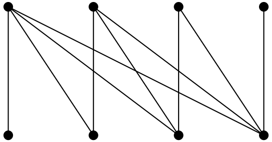
A class is *monadically dependent*, if it does not transduce the class of all graphs.

Equivalently: it does not transduce the class of all 1-subdivided bicliques.

Monadic Stability and Monadic Dependence

Definition

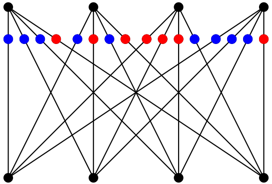
A class is *monadically stable*, if it does not transduce the class of all half graphs.



Definition

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

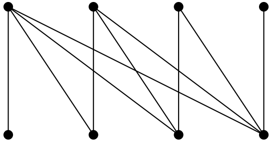
Equivalently: it does not transduce the class of all 1-subdivided bicliques.



Monadic Stability and Monadic Dependence

Definition

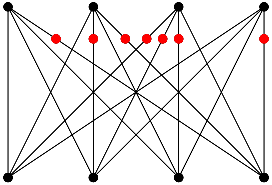
A class is *monadically stable*, if it does not transduce the class of all half graphs.



Definition

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

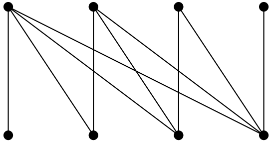
Equivalently: it does not transduce the class of all 1-subdivided bicliques.



Monadic Stability and Monadic Dependence

Definition

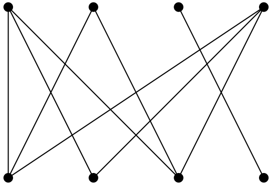
A class is *monadically stable*, if it does not transduce the class of all half graphs.



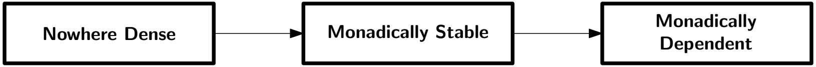
Definition

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

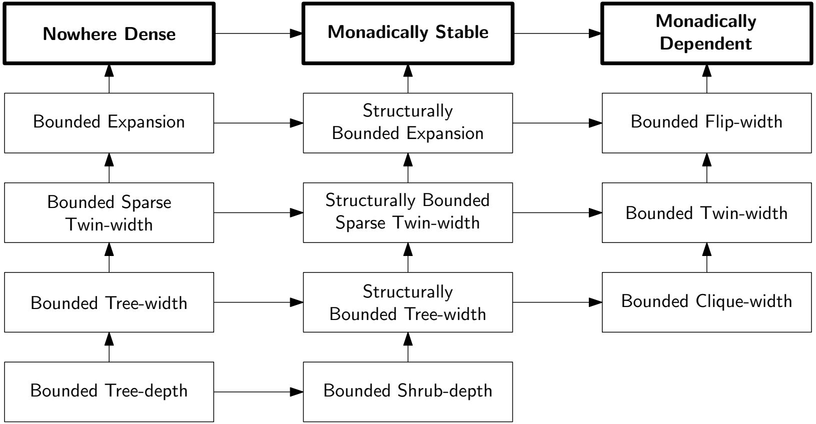
Equivalently: it does not transduce the class of all 1-subdivided bicliques.



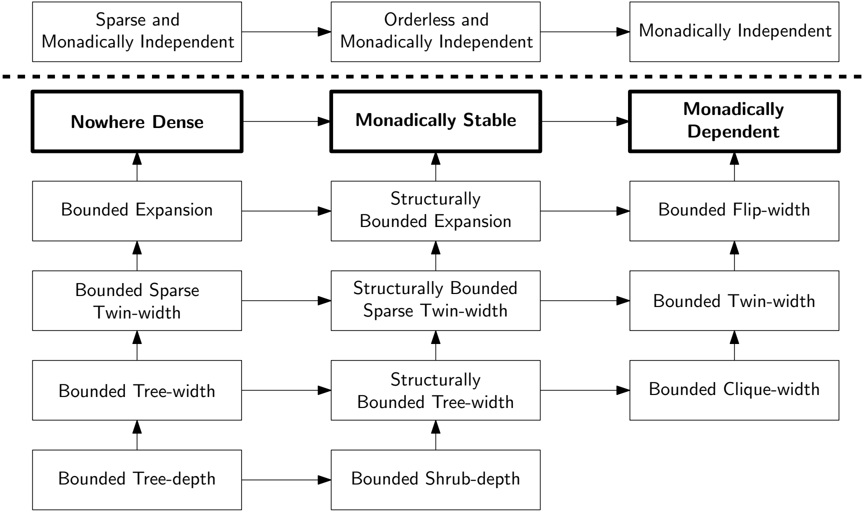
Model Checking in Hereditary Graph Classes



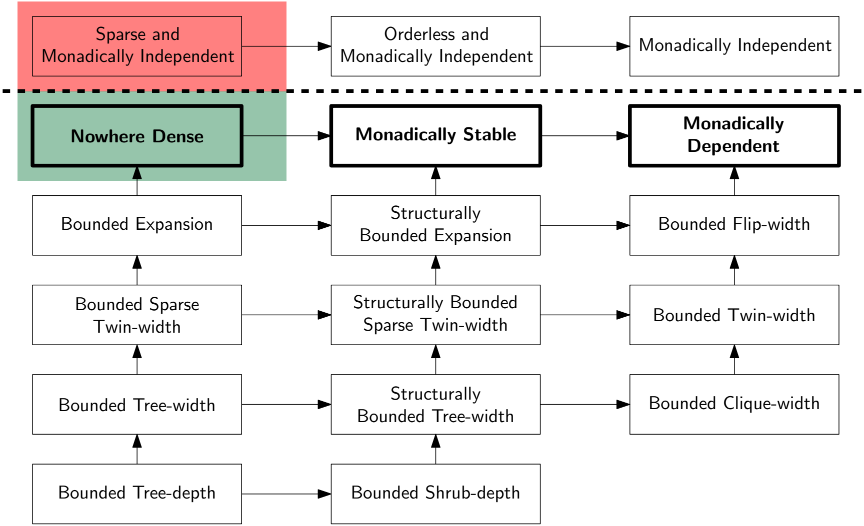
Model Checking in Hereditary Graph Classes



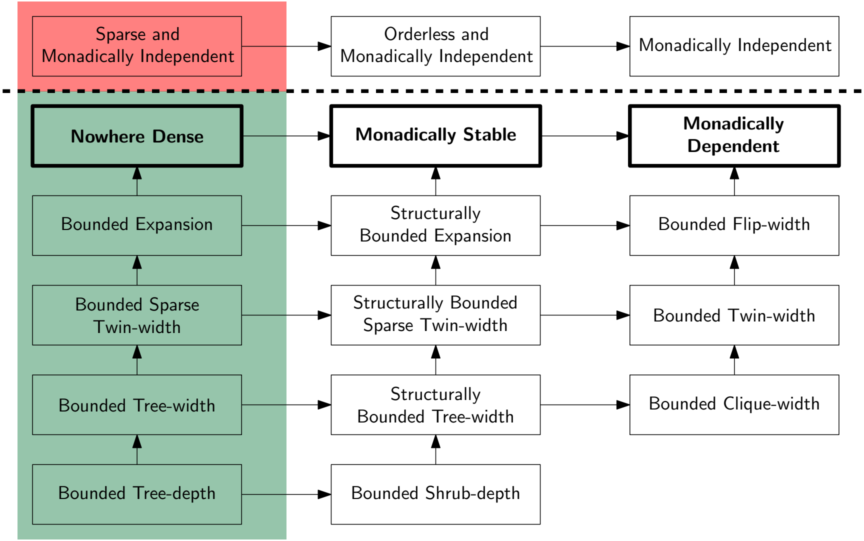
Model Checking in Hereditary Graph Classes



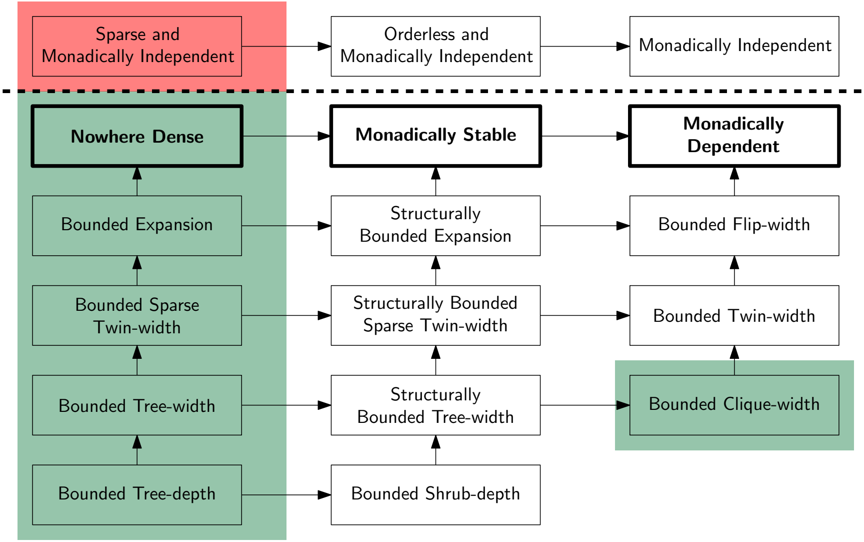
Model Checking in Hereditary Graph Classes



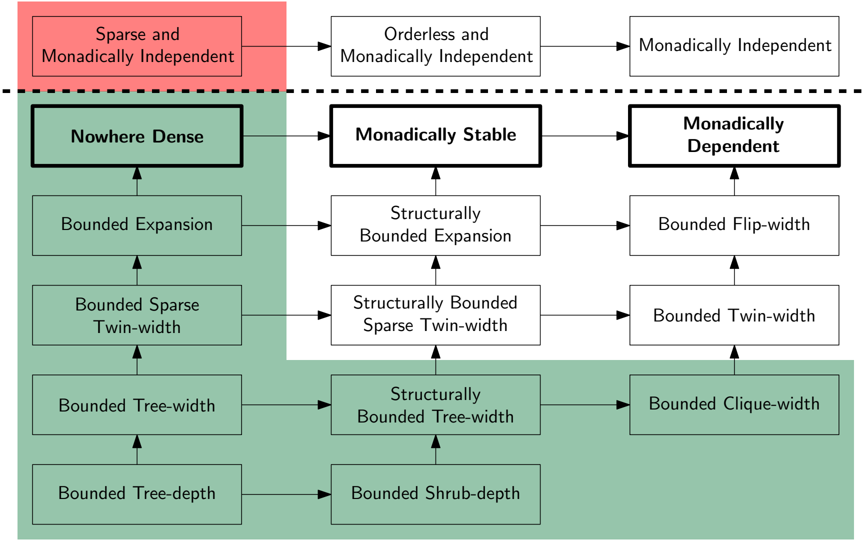
Model Checking in Hereditary Graph Classes



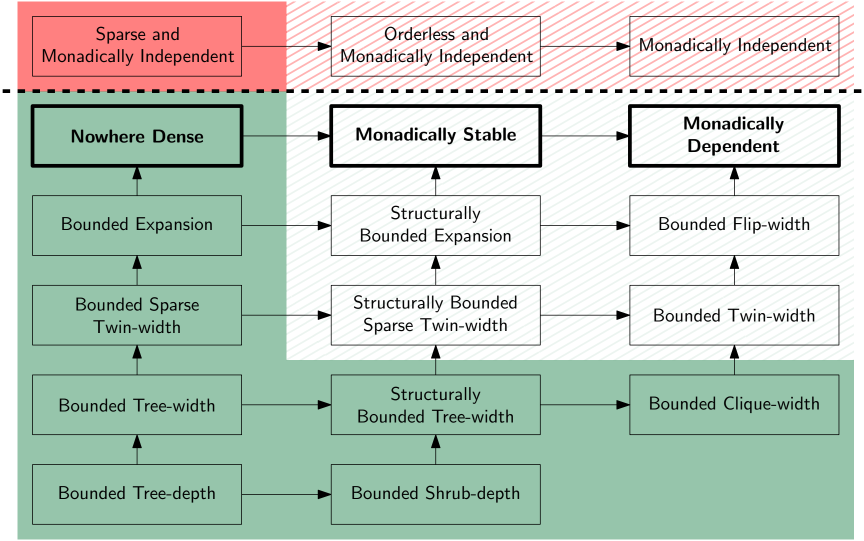
Model Checking in Hereditary Graph Classes



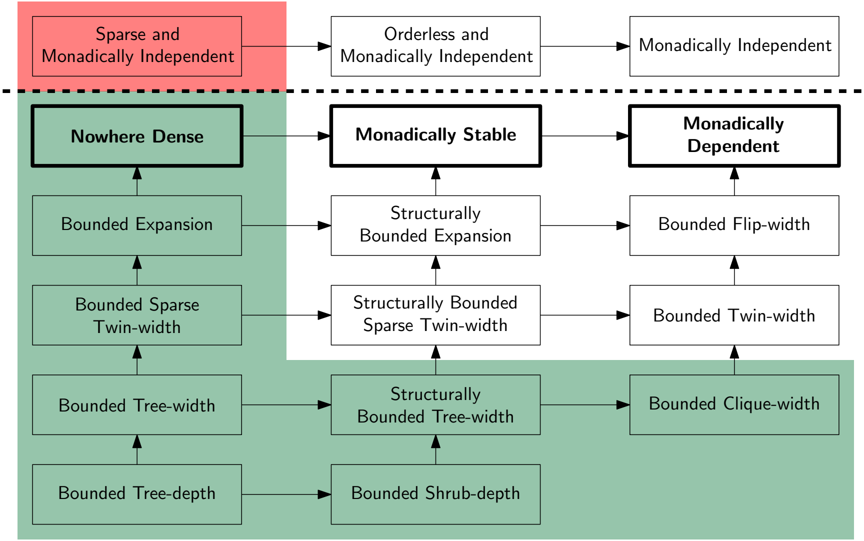
Model Checking in Hereditary Graph Classes



Conjectured Tractability Limits

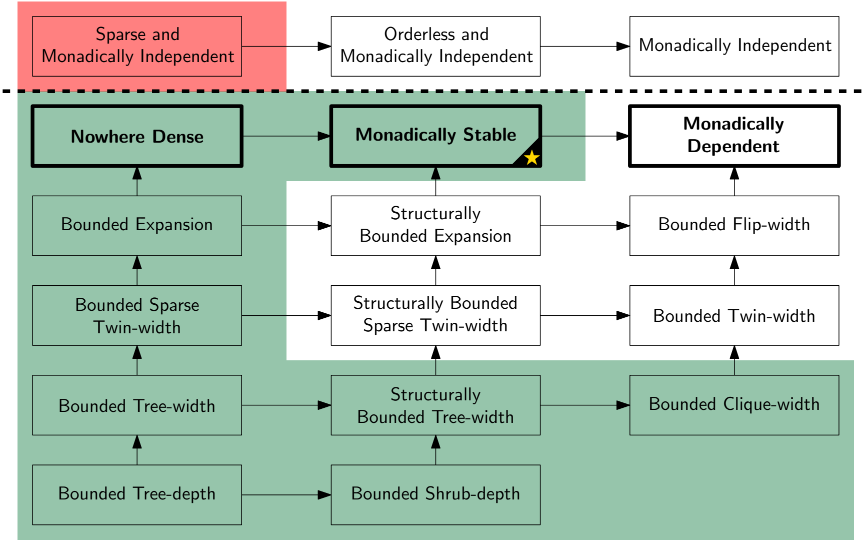


Our Results ★



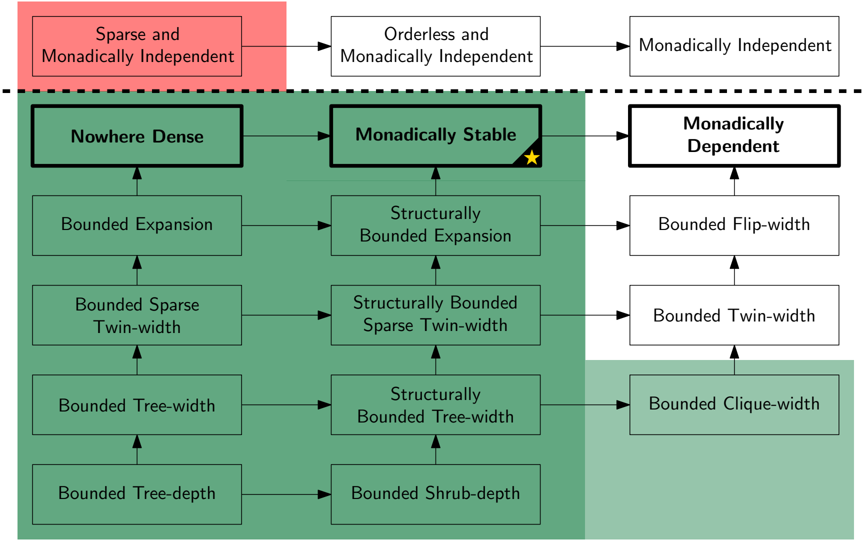
Sparse and Monadically Independent
Orderless and Monadically Independent
Monadically Independent
Nowhere Dense
Monadically Stable
Monadically Dependent
Bounded Expansion
Structurally Bounded Expansion
Bounded Flip-width
Bounded Sparse Twin-width
Structurally Bounded Sparse Twin-width
Bounded Twin-width
Bounded Tree-width
Structurally Bounded Tree-width
Bounded Clique-width
Bounded Tree-depth
Bounded Shrub-depth

Our Results ★



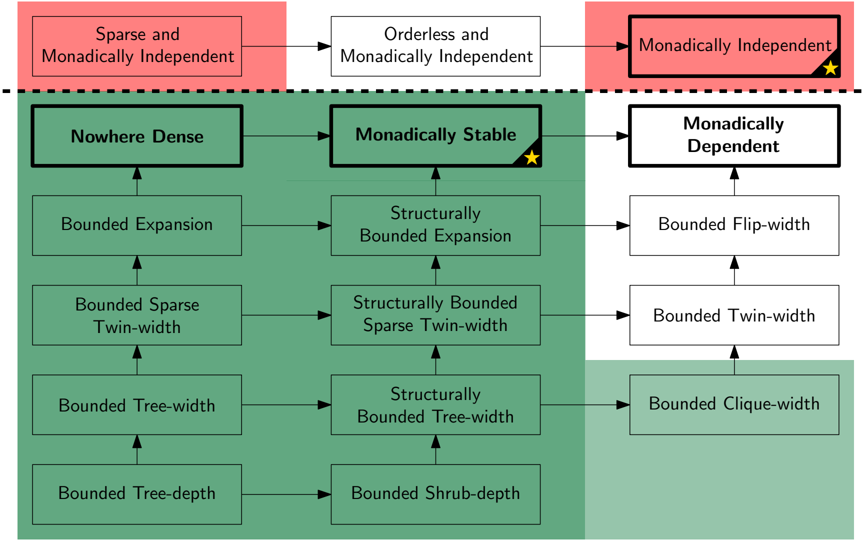
Sparse and Monadically Independent
Orderless and Monadically Independent
Monadically Independent
Nowhere Dense
Monadically Stable
Monadically Dependent
Bounded Expansion
Structurally Bounded Expansion
Bounded Flip-width
Bounded Sparse Twin-width
Structurally Bounded Sparse Twin-width
Bounded Twin-width
Bounded Tree-width
Structurally Bounded Tree-width
Bounded Clique-width
Bounded Tree-depth
Bounded Shrub-depth

Our Results ★



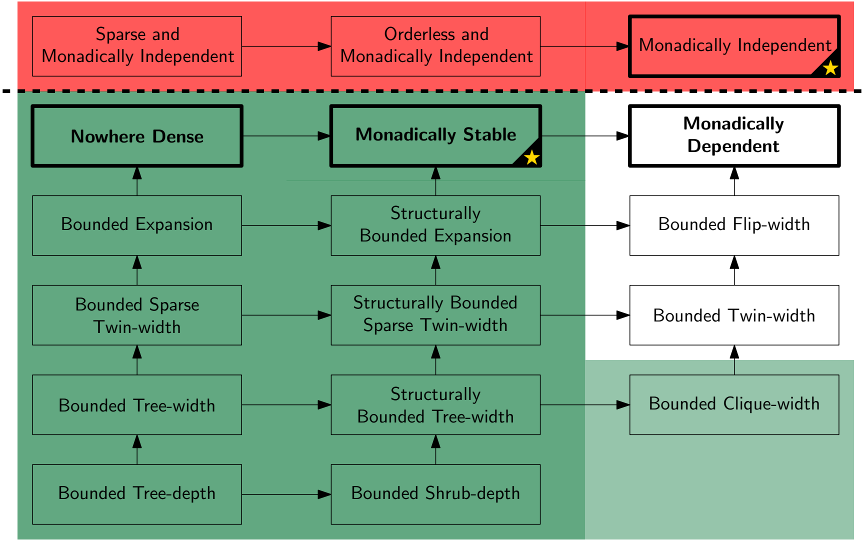
Sparse and Monadically Independent
Orderless and Monadically Independent
Monadically Independent
Nowhere Dense
Monadically Stable
Monadically Dependent
Bounded Expansion
Structurally Bounded Expansion
Bounded Flip-width
Bounded Sparse Twin-width
Structurally Bounded Sparse Twin-width
Bounded Twin-width
Bounded Tree-width
Structurally Bounded Tree-width
Bounded Clique-width
Bounded Tree-depth
Bounded Shrub-depth

Our Results ★



Sparse and Monadically Independent
Orderless and Monadically Independent
Monadically Independent
Nowhere Dense
Monadically Stable
Monadically Dependent
Bounded Expansion
Structurally Bounded Expansion
Bounded Flip-width
Bounded Sparse Twin-width
Structurally Bounded Sparse Twin-width
Bounded Twin-width
Bounded Tree-width
Structurally Bounded Tree-width
Bounded Clique-width
Bounded Tree-depth
Bounded Shrub-depth

Our Results ★


Sparse and Monadically Independent
Orderless and Monadically Independent
Monadically Independent
Nowhere Dense
Monadically Stable
Monadically Dependent
Bounded Expansion
Structurally Bounded Expansion
Bounded Flip-width
Bounded Sparse Twin-width
Structurally Bounded Sparse Twin-width
Bounded Twin-width
Bounded Tree-width
Structurally Bounded Tree-width
Bounded Clique-width
Bounded Tree-depth
Bounded Shrub-depth

Algorithmic Results

Theorem

There is a model checking algorithm with the following property.

For every monadically stable class $\mathcal{C}$, there exists a function $f : \mathbb{N} \times \mathbb{R} \to \mathbb{N}$ such that for every n-vertex graph $G \in \mathcal{C}$, sentence φ, and $\varepsilon > 0$, the algorithm runs in time

$$f(|\varphi|, \varepsilon) \cdot n^{6+\varepsilon}.$$

Theorem

Model checking is $\mathrm{AW}[*]$-hard on every hereditary, monadically independent class.

Combinatorial Results

Monadic stability and dependence are defined through **logic**.

Algorithmic results require a **combinatorial** understanding.

Combinatorial Results

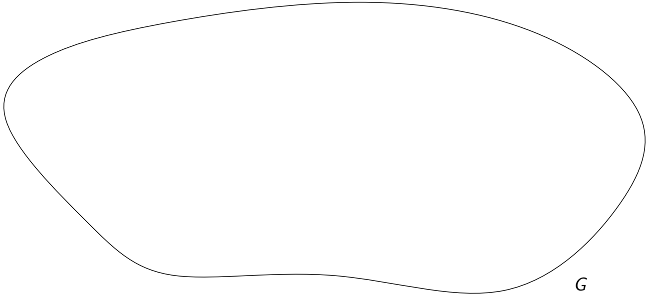
Monadic stability and dependence are defined through **logic**.

Algorithmic results require a **combinatorial** understanding.

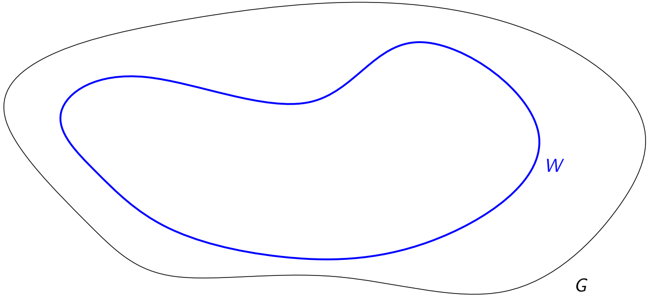
Main part of the thesis: combinatorial characterizations of mon. stability/dependence

- Ramsey-theoretic characterizations
- Forbidden induced subgraphs characterizations
- Game characterization (only for monadic stability)

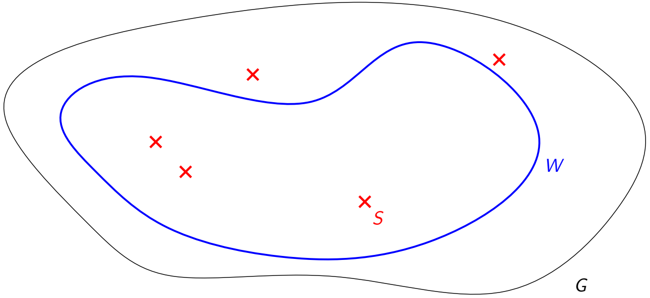
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



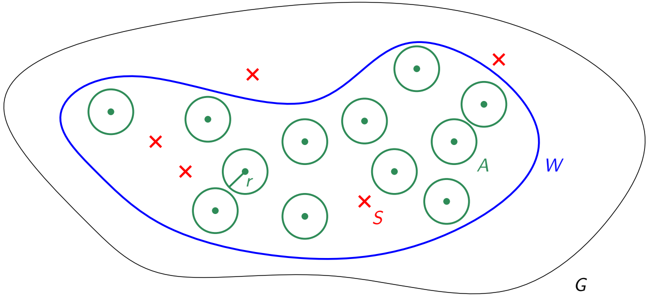
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



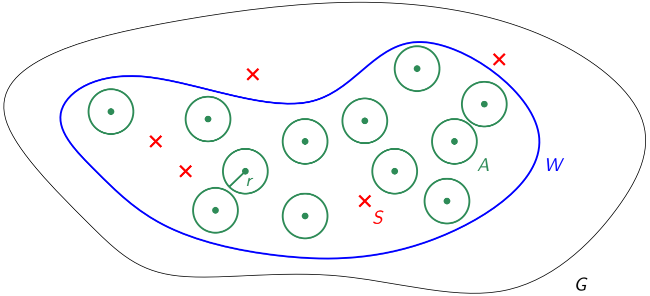
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



Characterizing Nowhere Denseness: Uniform Quasi-Wideness



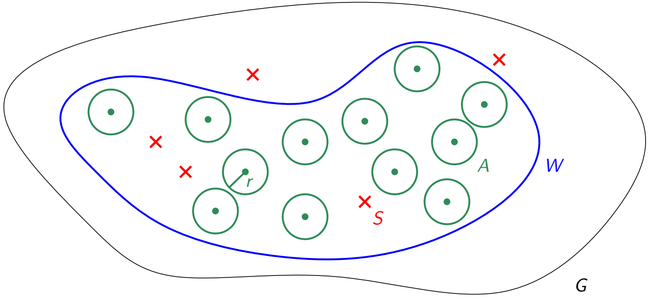
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



Uniform Quasi-Wideness (slightly informal)

A class $\mathcal{C}$ is *uniformly quasi-wide* if for every radius r, in every large set W we find a still large set A that is r-independent after removing a set S of constantly many vertices.

Characterizing Nowhere Denseness: Uniform Quasi-Wideness



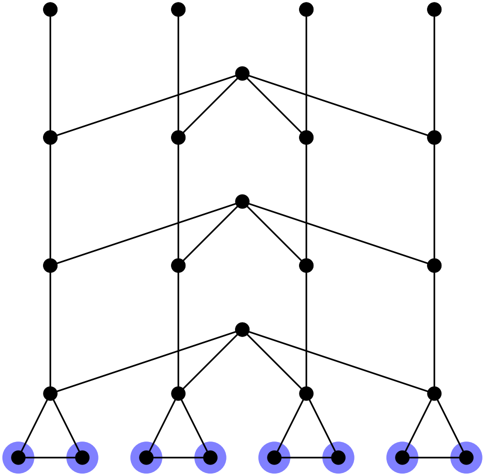
Uniform Quasi-Wideness (slightly informal)

A class $\mathcal{C}$ is *uniformly quasi-wide* if for every radius r, in every large set W we find a still large set A that is r-independent after removing a set S of constantly many vertices.

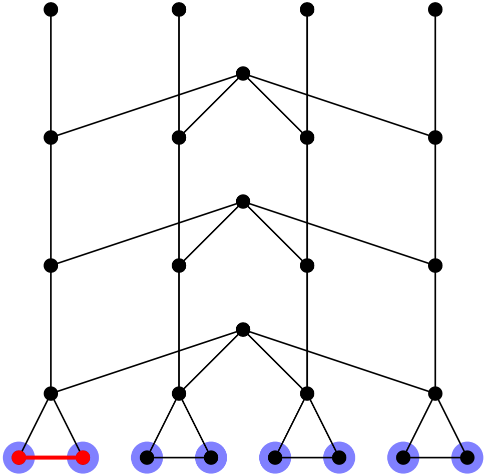
Theorem [Něsetřil, Ossona de Mendez, 2011]

A class $\mathcal{C}$ is uniformly quasi-wide if and only if it is nowhere dense.

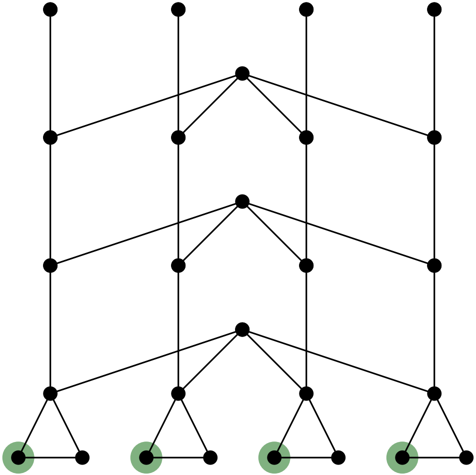
Uniform Quasi-Wideness: Example



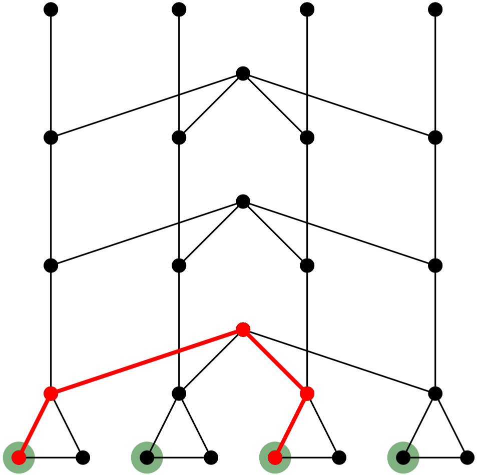
Uniform Quasi-Wideness: Example



Uniform Quasi-Wideness: Example

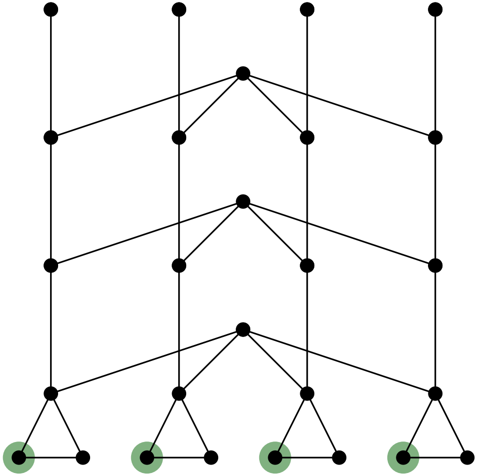


Uniform Quasi-Wideness: Example

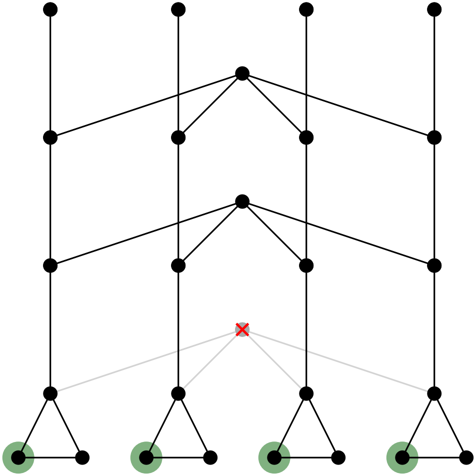


dist < 4

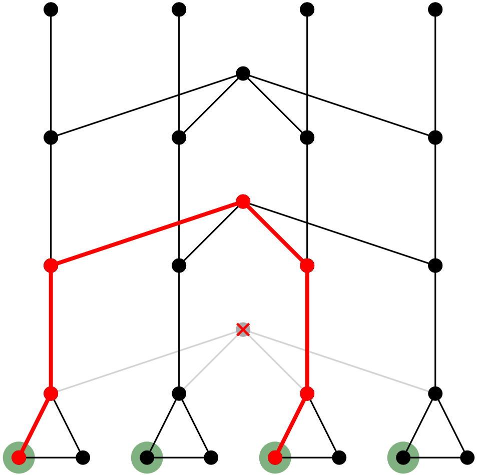
Uniform Quasi-Wideness: Example



Uniform Quasi-Wideness: Example

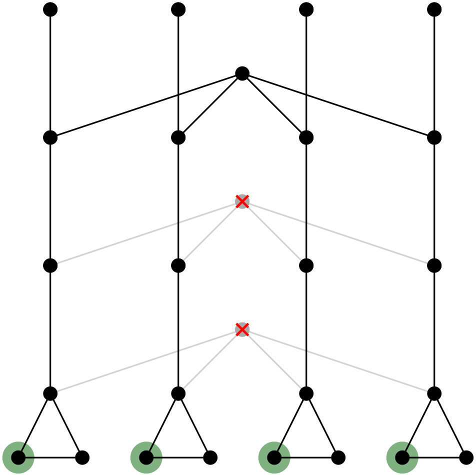


Uniform Quasi-Wideness: Example

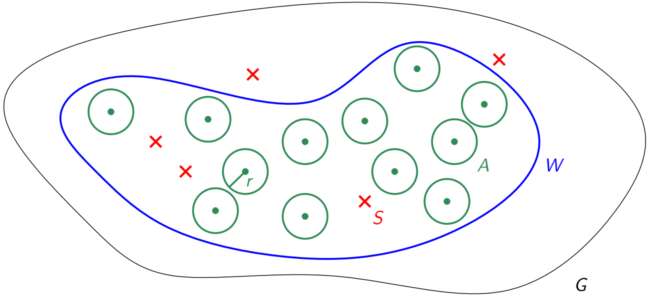


dist < 6

Uniform Quasi-Wideness: Example



Characterizing Nowhere Denseness: Uniform Quasi-Wideness



Uniform Quasi-Wideness (slightly informal)

A class $\mathcal{C}$ is *uniformly quasi-wide* if for every radius r, in every large set W we find a still large set A that is r-independent after removing a set S of constantly many vertices.

Theorem [Něsetřil, Ossona de Mendez, 2011]

A class $\mathcal{C}$ is uniformly quasi-wide if and only if it is nowhere dense.

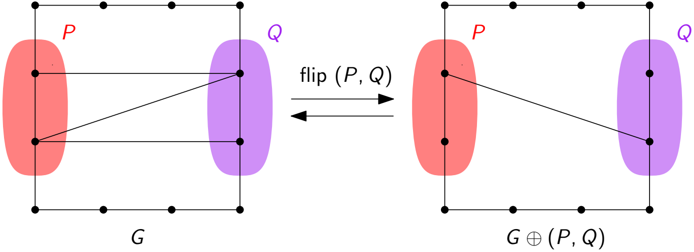
Towards Dense Graphs

Question: Is there a similar characterization for monadic stability/dependence?

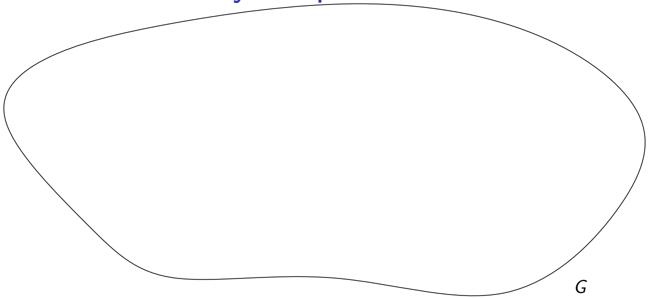
Towards Dense Graphs

Question: Is there a similar characterization for monadic stability/dependence?

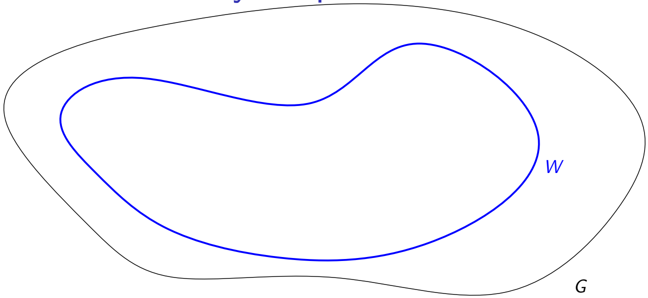
Denote by $G \oplus (P, Q)$ the graph obtained from G by complementing edges between pairs of vertices from $P \times Q$.



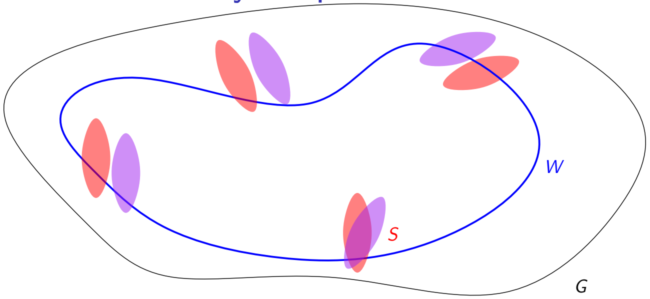
Characterizing Monadic Stability: Flip-Flatness



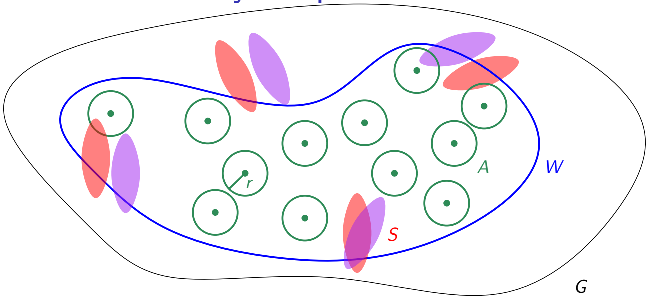
Characterizing Monadic Stability: Flip-Flatness



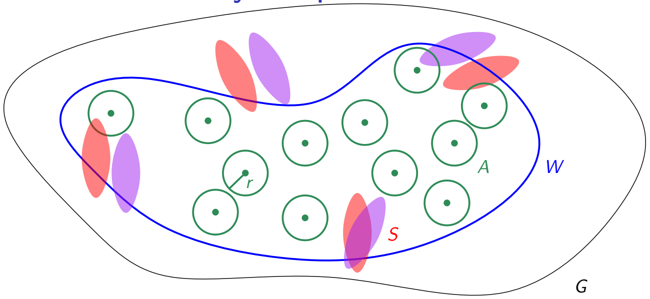
Characterizing Monadic Stability: Flip-Flatness



Characterizing Monadic Stability: Flip-Flatness



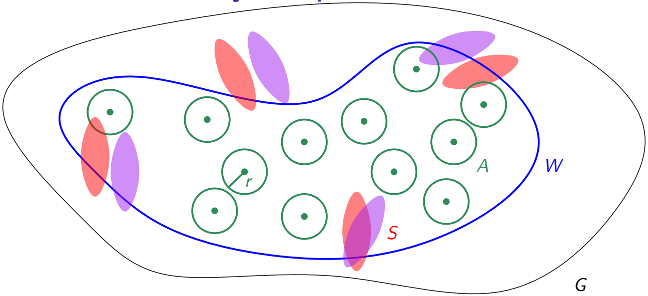
Characterizing Monadic Stability: Flip-Flatness



Flip-Flatness (slightly informal)

A class $\mathcal{C}$ is *flip-flat* if for every radius r, in every large set W we find a still large set A that is r-independent after performing a set S of constantly many flips.

Characterizing Monadic Stability: Flip-Flatness



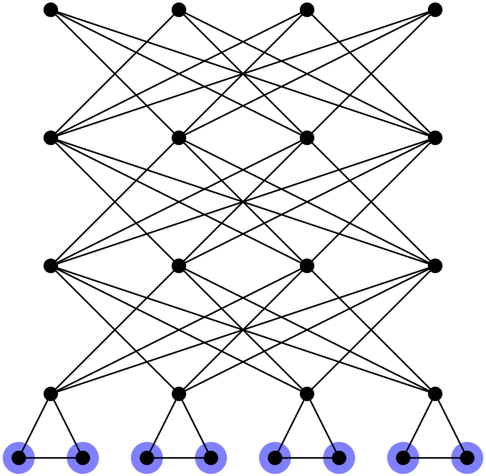
Flip-Flatness (slightly informal)

A class $\mathcal{C}$ is *flip-flat* if for every radius r, in every large set W we find a still large set A that is r-independent after performing a set S of constantly many flips.

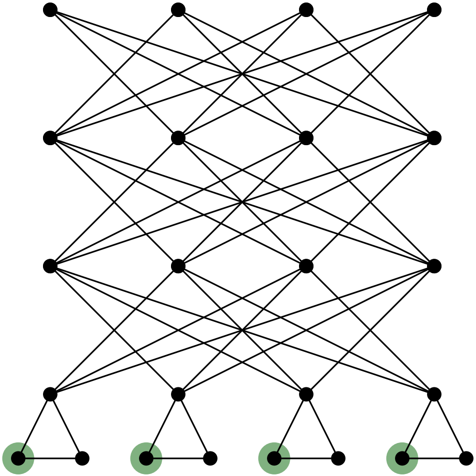
Theorem

A class $\mathcal{C}$ is flip-flat if and only if it is monadically stable.

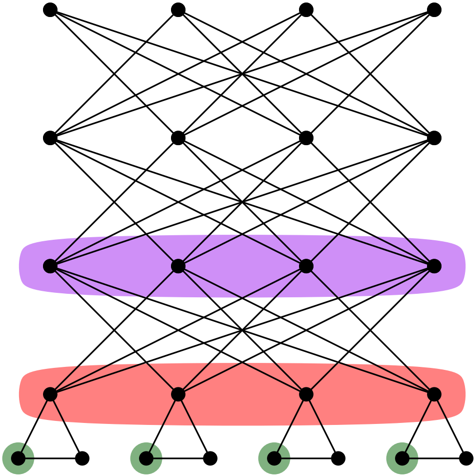
Flip-Flatness: Example



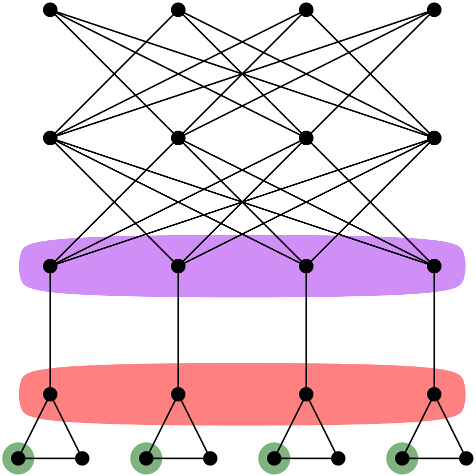
Flip-Flatness: Example



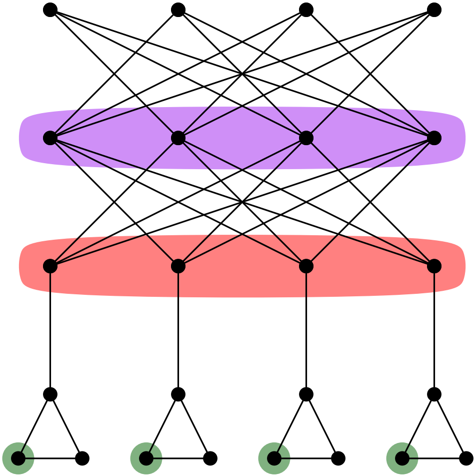
Flip-Flatness: Example



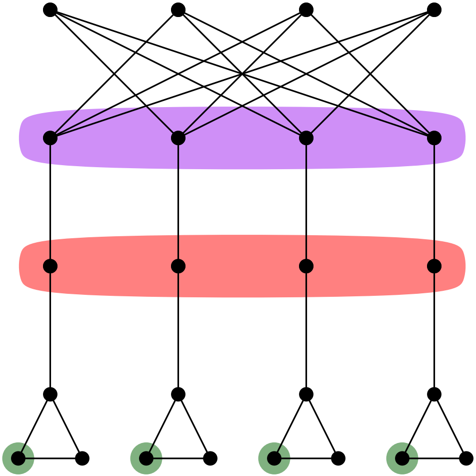
Flip-Flatness: Example



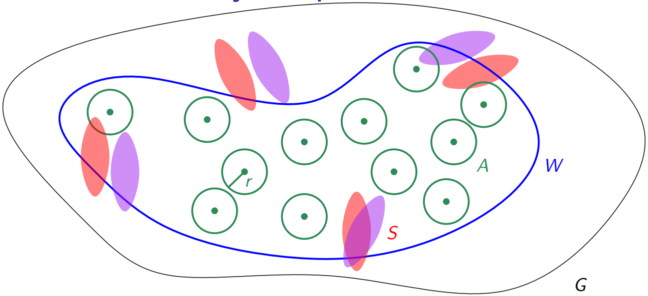
Flip-Flatness: Example



Flip-Flatness: Example



Characterizing Monadic Stability: Flip-Flatness



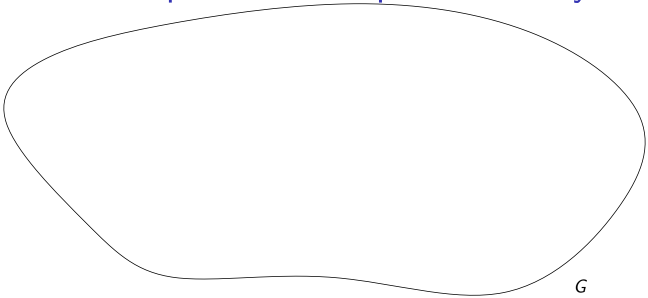
Flip-Flatness (slightly informal)

A class $\mathcal{C}$ is *flip-flat* if for every radius r, in every large set W we find a still large set A that is r-independent after performing a set S of constantly many flips.

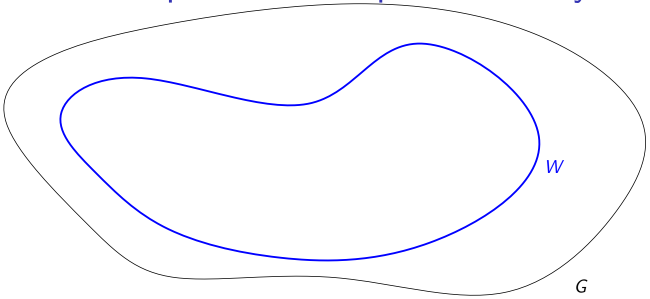
Theorem

A class $\mathcal{C}$ is flip-flat if and only if it is monadically stable.

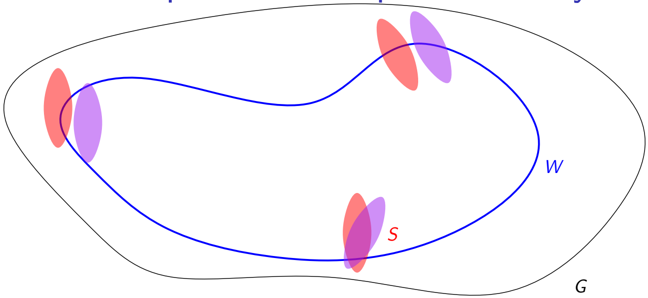
Characterizing Monadic Dependence: Flip-Breakability



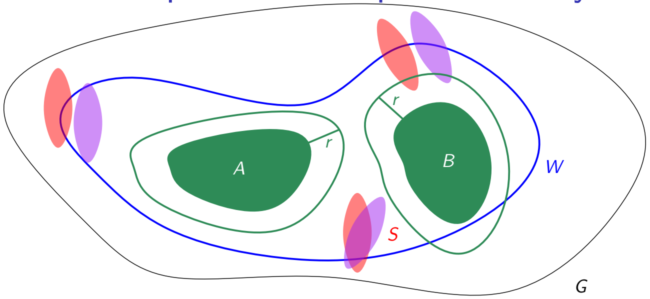
Characterizing Monadic Dependence: Flip-Breakability



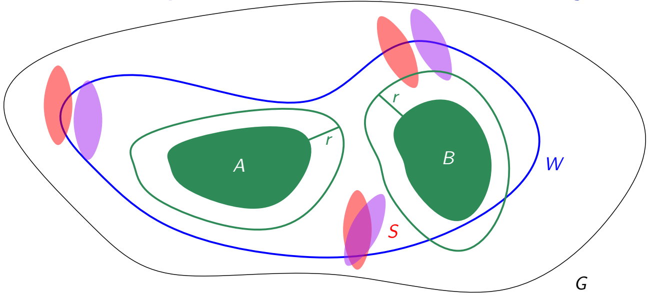
Characterizing Monadic Dependence: Flip-Breakability



Characterizing Monadic Dependence: Flip-Breakability



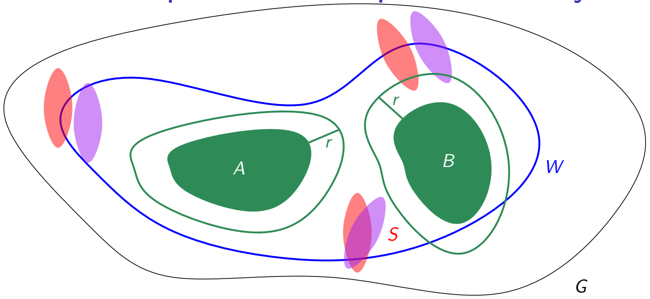
Characterizing Monadic Dependence: Flip-Breakability



Flip-Breakability (slightly informal)

A class $\mathcal{C}$ is *flip-breakable* if for every radius r, in every large set W we find two large sets A and B that are at distance greater than $2r$ from each other after performing a set S of constantly many flips.

Characterizing Monadic Dependence: Flip-Breakability



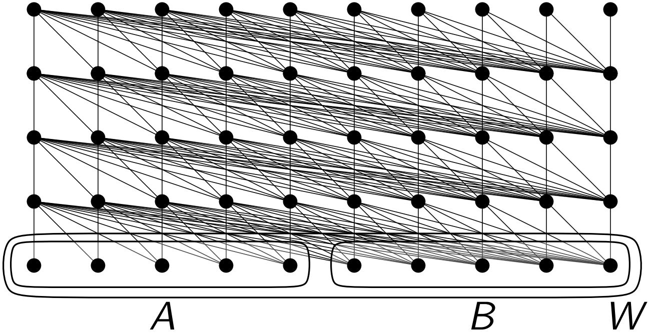
Flip-Breakability (slightly informal)

A class $\mathcal{C}$ is *flip-breakable* if for every radius r, in every large set W we find two large sets A and B that are at distance greater than $2r$ from each other after performing a set S of constantly many flips.

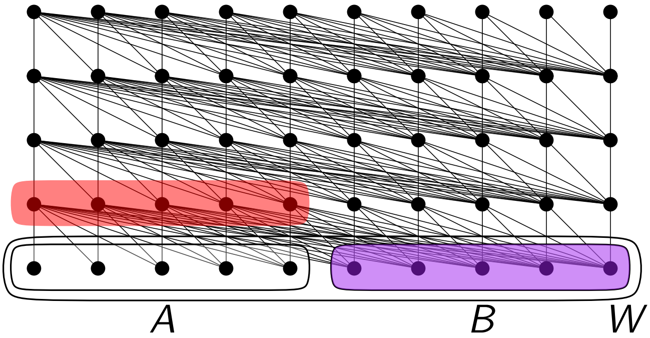
Theorem

A class $\mathcal{C}$ is flip-breakable if and only if it is monadically dependent.

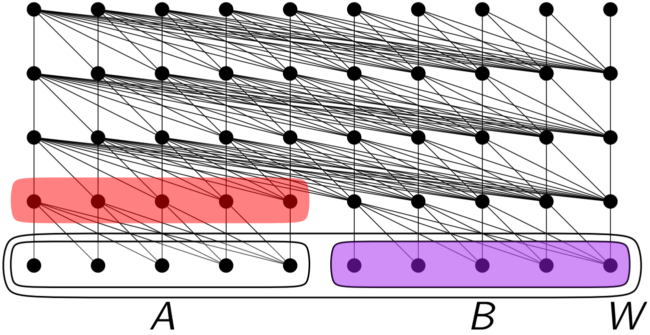
Flip-Breakability: Example



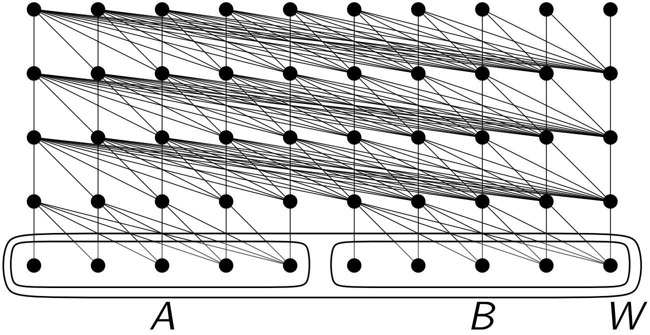
Flip-Breakability: Example



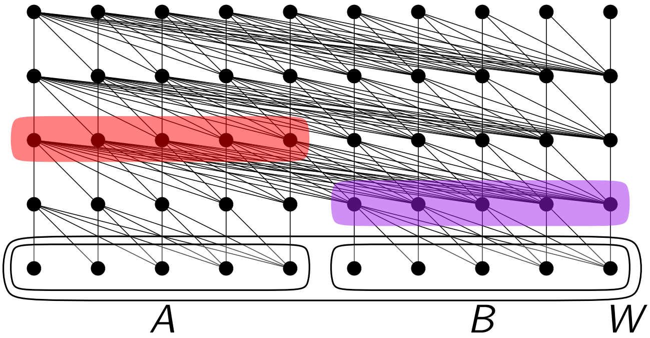
Flip-Breakability: Example



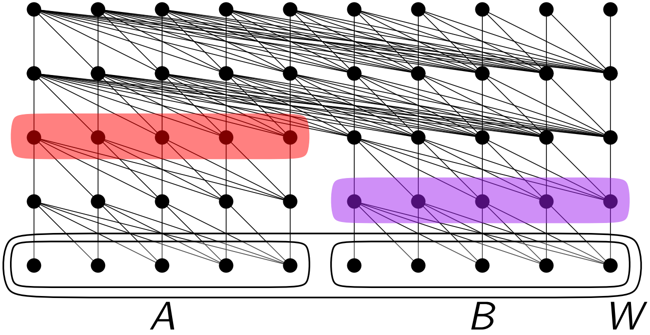
Flip-Breakability: Example



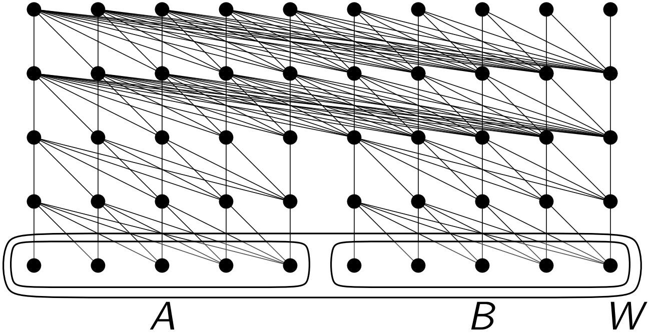
Flip-Breakability: Example



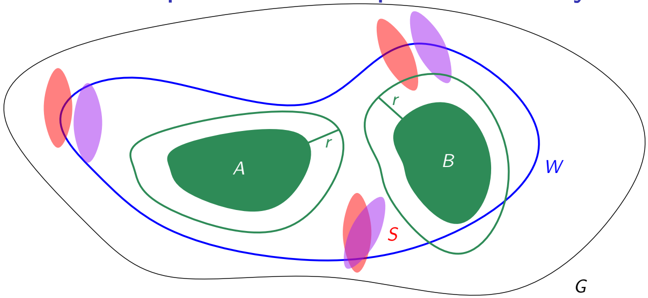
Flip-Breakability: Example



Flip-Breakability: Example



Characterizing Monadic Dependence: Flip-Breakability



Flip-Breakability (slightly informal)

A class $\mathcal{C}$ is *flip-breakable* if for every radius r, in every large set W we find two large sets A and B that are at distance greater than $2r$ from each other after performing a set S of constantly many flips.

Theorem

A class $\mathcal{C}$ is flip-breakable if and only if it is monadically dependent.

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

		flatness	breakability
dist-r	flip-	**monadic stability**	**monadic dependence**
	deletion-	**nowhere denseness**	
dist-∞	flip-		
	deletion-		

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

		flatness	breakability
dist-r	flip-	**monadic stability**	**monadic dependence**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-		
	deletion-		

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

		flatness	breakability
dist-r	flip-	**monadic stability**	**monadic dependence**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-	**bd. shrub-depth**	**bd. clique-width**
	deletion-		

Variants of Flip-Breakability

1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

		flatness	breakability
dist-r	flip-	**monadic stability**	**monadic dependence**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-	**bd. shrub-depth**	**bd. clique-width**
	deletion-	**bd. tree-depth**	**bd. tree-width**

Variants of Flip-Breakability

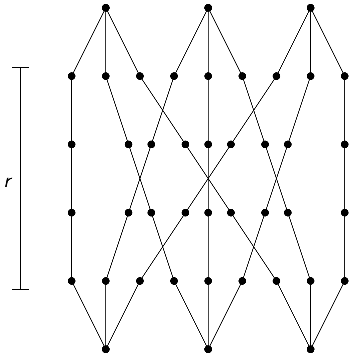
1. We modify a graph using either flips or vertex deletions.
2. We demand our resulting set is either flat or broken.

   flat: pairwise separated; broken: separated into two large sets
3. Separation means either distance-r or distance-∞.

		flatness	breakability
dist-r	flip-	**monadic stability**	**monadic dependence**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-	**bd. shrub-depth**	**bd. clique-width**
	deletion-	**bd. tree-depth**	**bd. tree-width**

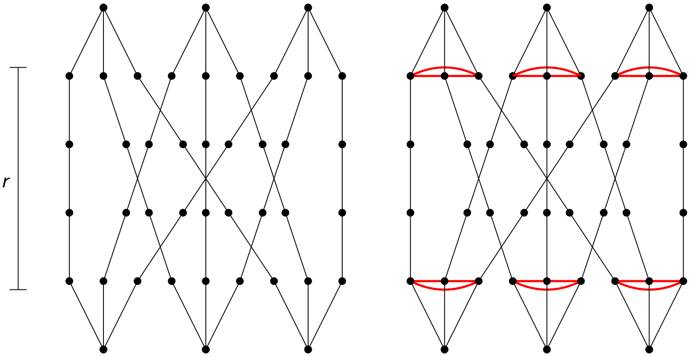
Ramsey-theoretic characterization ✓ next up: forbidden induced subgraphs

Characterizing Monadic Dependence by Forbidden Induced Subgraphs



star r-crossing
$= r$-subdivided biclique

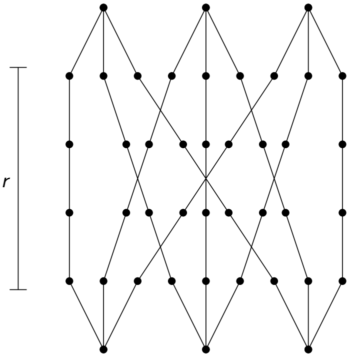
Characterizing Monadic Dependence by Forbidden Induced Subgraphs



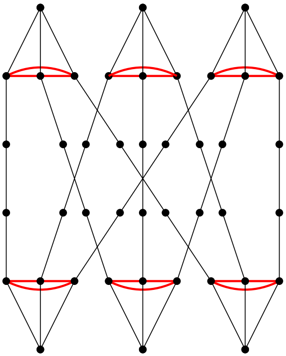
star r-crossing
$= r$-subdivided biclique

clique r-crossing

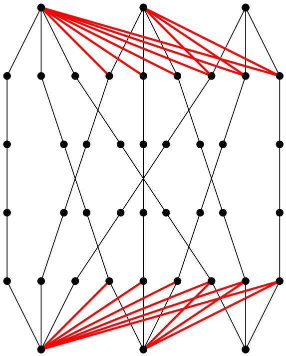
Characterizing Monadic Dependence by Forbidden Induced Subgraphs



star r-crossing
$= r$-subdivided biclique

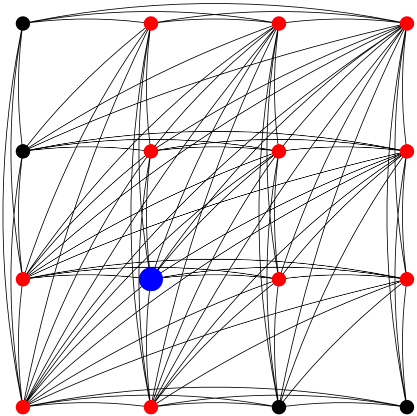


clique r-crossing



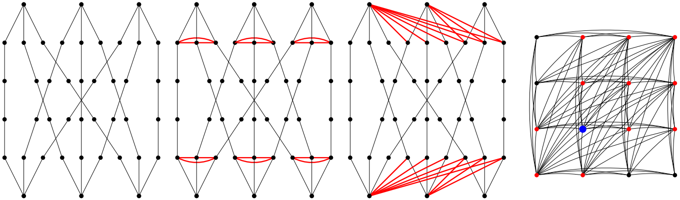
half-graph r-crossing

Characterizing Monadic Dependence by Forbidden Induced Subgraphs



comparability grid

Characterizing Monadic Dependence by Forbidden Induced Subgraphs

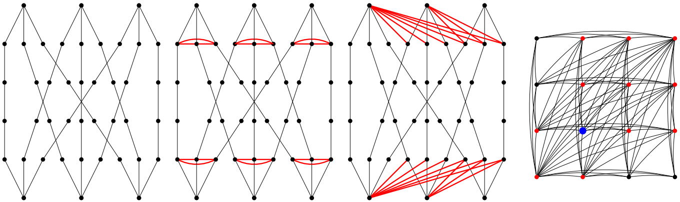


Theorem

Let $\mathcal{C}$ be a graph class. Then $\mathcal{C}$ is monadically dependent if and only if for every $r \geq 1$ there exists $k \in \mathbb{N}$ such $\mathcal{C}$ excludes as induced subgraphs

- all layerwise flipped star r-crossings of order k, and
- all layerwise flipped clique r-crossings of order k, and
- all layerwise flipped half-graph r-crossings of order k, and
- the comparability grid of order k.

Characterizing Monadic Dependence by Forbidden Induced Subgraphs



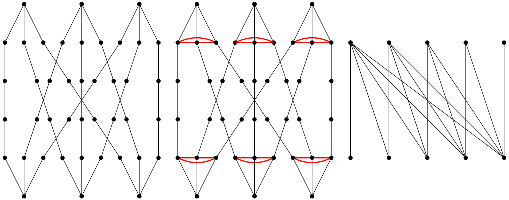
Theorem

Let $\mathcal{C}$ be a graph class. Then $\mathcal{C}$ is monadically dependent if and only if for every $r \geq 1$ there exists $k \in \mathbb{N}$ such $\mathcal{C}$ excludes as induced subgraphs

- all layerwise flipped star r-crossings of order k, and
- all layerwise flipped clique r-crossings of order k, and
- all layerwise flipped half-graph r-crossings of order k, and
- the comparability grid of order k.

$\Rightarrow$ Model checking is hard on every hereditary, monadically independent graph class.

Characterizing Monadic Stability by Forbidden Induced Subgraphs

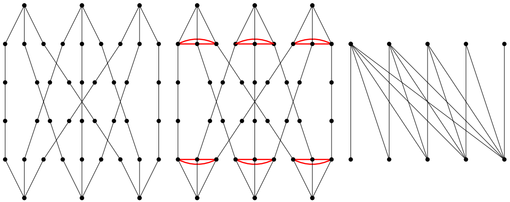


Theorem

Let $\mathcal{C}$ be a graph class. Then $\mathcal{C}$ is monadically stable if and only if for every $r \geq 1$ there exists $k \in \mathbb{N}$ such $\mathcal{C}$ excludes as induced subgraphs

- all layerwise flipped star r-crossings of order k, and
- all layerwise flipped clique r-crossings of order k, and
- all semi-induced halfgraphs of order k

Characterizing Monadic Stability by Forbidden Induced Subgraphs



Theorem

Let $\mathcal{C}$ be a graph class. Then $\mathcal{C}$ is monadically stable if and only if for every $r \geq 1$ there exists $k \in \mathbb{N}$ such $\mathcal{C}$ excludes as induced subgraphs

- all layerwise flipped star r-crossings of order k, and
- all layerwise flipped clique r-crossings of order k, and
- all semi-induced halfgraphs of order k

Characterizations: ramsey-theoretic ✓ forbidden induced subgraphs ✓
Next up: a game characterization for monadic stability

The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

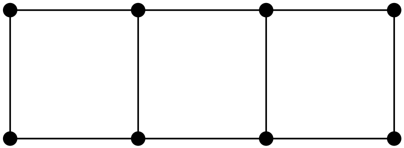
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



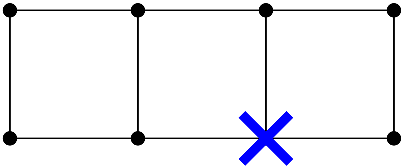
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



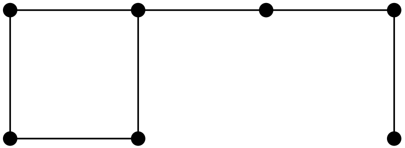
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



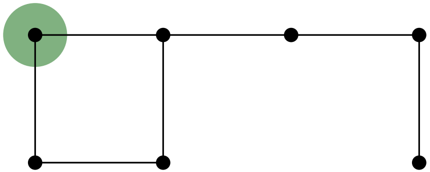
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



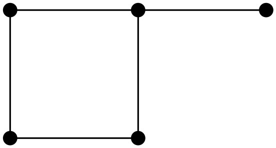
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



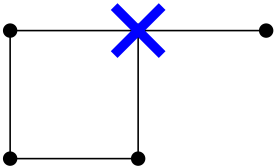
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



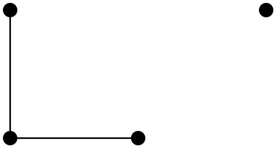
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



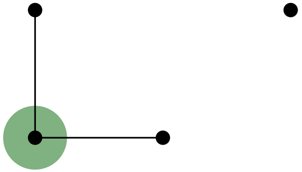
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



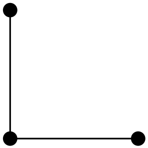
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



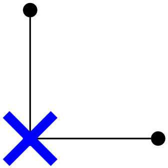
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



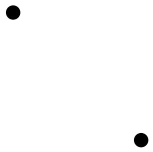
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



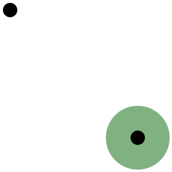
The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:



The Splitter Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v to delete
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

Example play of the radius-2 Splitter game:

The Splitter Game in Nowhere Dense Classes

Theorem [Grohe, Kreutzer, Siebertz, 2013]

A graph class $\mathcal{C}$ is nowhere dense $\Leftrightarrow$

$\forall r \exists \ell$ such that Splitter wins the radius-r game on all graphs from $\mathcal{C}$ in ℓ rounds.

The Splitter Game in Nowhere Dense Classes

Theorem [Grohe, Kreutzer, Siebertz, 2013]

A graph class $\mathcal{C}$ is nowhere dense $\Leftrightarrow$

$\forall r \exists \ell$ such that Splitter wins the radius-r game on all graphs from $\mathcal{C}$ in ℓ rounds.

Question: Can we find a similar **game characterization** for monadic stability?

The Flipper Game

The radius-r Splitter game is played on a graph G_1. In round i

1. Splitter chooses a vertex v
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

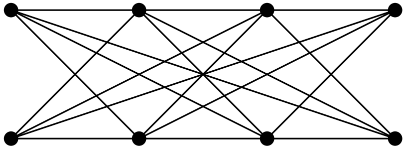
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



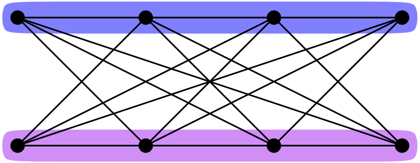
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



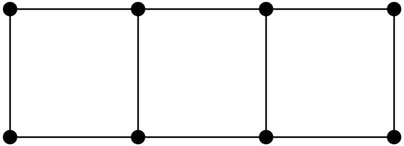
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



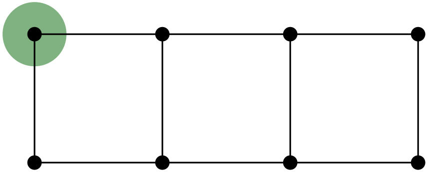
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



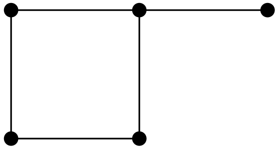
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



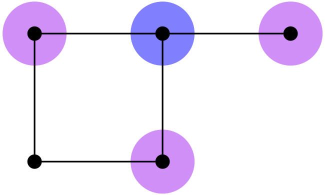
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



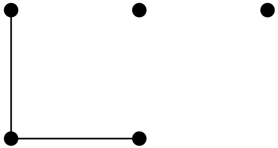
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



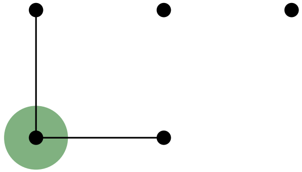
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



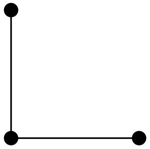
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



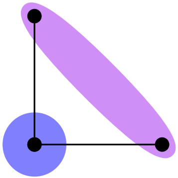
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



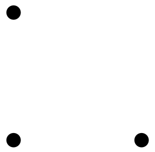
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



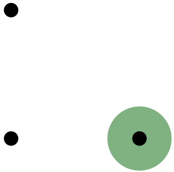
The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:



The Flipper Game

The radius-r Flipper game is played on a graph G_1. In round i

1. Flipper chooses a flip F
2. Localizer chooses G_{i+1} as a radius-r ball in $G_i \oplus F$.

Flipper wins once G_i has size 1.

Example play of the radius-2 Flipper game:

The Flipper Game in Monadically Stable Classes

Theorem

A graph class $\mathcal{C}$ is monadically stable $\Leftrightarrow$

$\forall r \exists \ell$ such that Flipper wins the radius-r game on all graphs from $\mathcal{C}$ in ℓ rounds.

Proof builds on flip-flatness. Flippers moves are computable in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.

The Flipper Game in Monadically Stable Classes

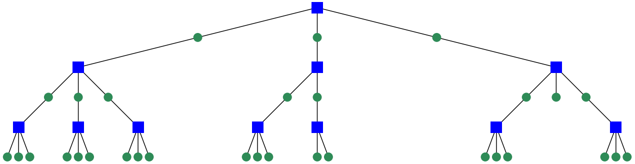
Theorem

A graph class $\mathcal{C}$ is monadically stable $\Leftrightarrow$

$\forall r \exists \ell$ such that Flipper wins the radius-r game on all graphs from $\mathcal{C}$ in ℓ rounds.

Proof builds on flip-flatness. Flippers moves are computable in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.

The game tree is a **bounded depth** decomposition of a graph into r-neighborhoods.



The Flipper Game in Monadically Stable Classes

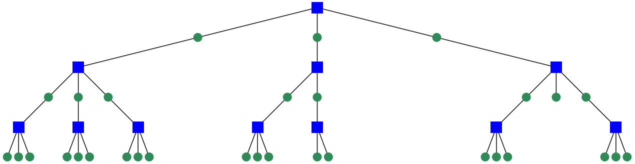
Theorem

A graph class $\mathcal{C}$ is monadically stable $\Leftrightarrow$

$\forall r \exists \ell$ such that Flipper wins the radius-r game on all graphs from $\mathcal{C}$ in ℓ rounds.

Proof builds on flip-flatness. Flippers moves are computable in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.

The game tree is a **bounded depth** decomposition of a graph into r-neighborhoods.

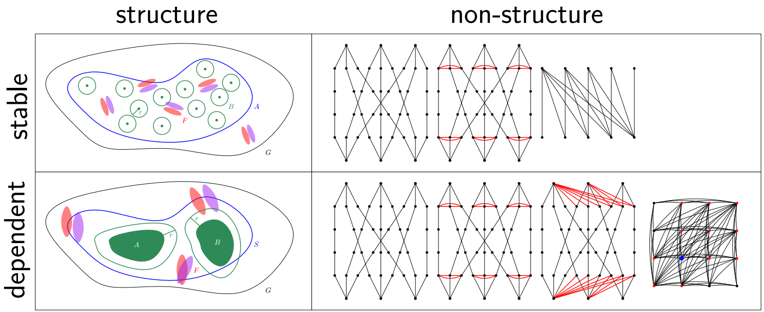


The decomposition can be further compressed by clustering neighborhoods.

Dynamic programming on the compressed tree gives **fpt model checking**.

Summary

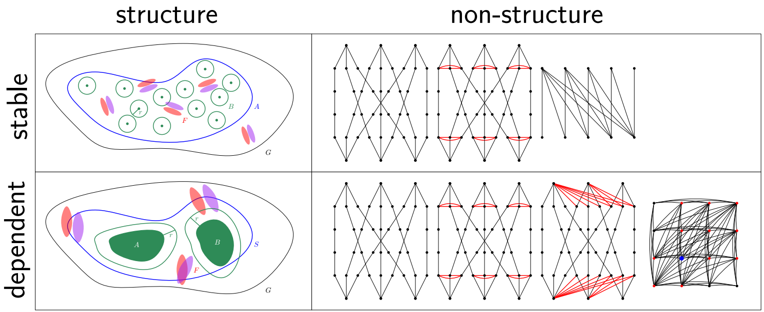
We have initiated the development of a combinatorial theory for monadically stable and dependent graph classes:



Algorithmic applications: model checking is fpt on every monadically stable class, but $\mathrm{AW}[*]$-hard on every hereditary, monadically independent class.

Summary

We have initiated the development of a combinatorial theory for monadically stable and dependent graph classes:



Algorithmic applications: model checking is fpt on every monadically stable class, but $\mathrm{AW}[*]$-hard on every hereditary, monadically independent class.

Vielen Dank!

Backup slides

Publications 1/2

1. *Indiscernibles and Flatness in Monadically Stable and Monadically NIP Classes*
   joint work with Jan Dreier, Sebastian Siebertz, Szymon Toruńczyk
   presented at ICALP 2023

2. *Flipper Games for Monadically Stable Graph Classes*
   joint work with Jakub Gajarský, Rose McCarty, Pierre Ohlmann, Michał Pilipczuk, Wojciech Przybyszewski, Sebastian Siebertz, Marek Sokołowski, Szymon Toruńczyk
   presented at ICALP 2023

3. *First-Order Model Checking on Structurally Sparse Graph Classes*
   joint work with Jan Dreier, Sebastian Siebertz
   presented at STOC 2023

Publications 2/2

4. *First-Order Model Checking on Monadically Stable Graph Classes*
   joint work with Jan Dreier, Ioannis Eleftheriadis, Rose McCarty, Michał Pilipczuk, Szymon Toruńczyk
   accepted at FOCS 2024

5. *Flip-Breakability: A Combinatorial Dichotomy for Monadically Dependent Graph Classes*
   joint work with Jan Dreier, Szymon Toruńczyk
   presented at STOC 2024

Flip-Flatness

Theorem

A graph class $\mathcal{C}$ is *flip-flat* if for every radius $r \in \mathbb{N}$ there exists a function $N_r : \mathbb{N} \to \mathbb{N}$ and a constant $k_r \in \mathbb{N}$ such that for all $m \in \mathbb{N}$, $G \in \mathcal{C}$ and $W \subseteq V(G)$ with $|W| \geq N_r(m)$ there exist a subset $A \subset W$ with $|A| \geq m$ and a k_r-flip H of G such that for every two distinct vertices $u, v \in A$:

$$\mathrm{dist}_H(u, v) > r.$$

Flip-Breakability

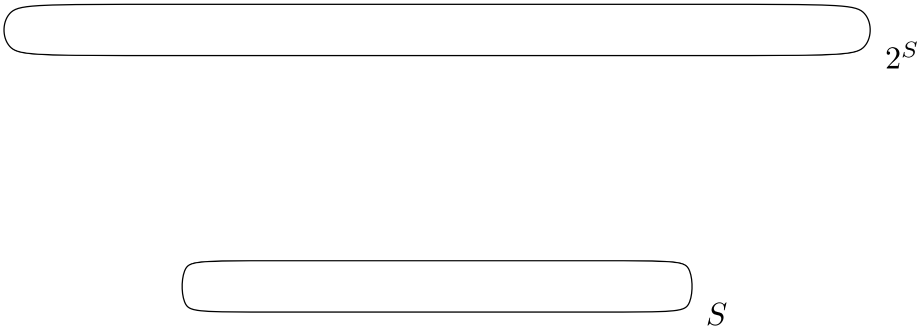
Theorem

A graph class $\mathcal{C}$ is *flip-breakable* if for every radius $r \in \mathbb{N}$ there exists a function $N_r : \mathbb{N} \to \mathbb{N}$ and a constant $k_r \in \mathbb{N}$ such that for all $m \in \mathbb{N}$, $G \in \mathcal{C}$ and $W \subseteq V(G)$ with $|W| \geq N_r(m)$ there exist subsets $A, B \subset W$ with $|A|, |B| \geq m$ and a k_r-flip H of G such that:

$$\mathrm{dist}_H(A, B) > r.$$

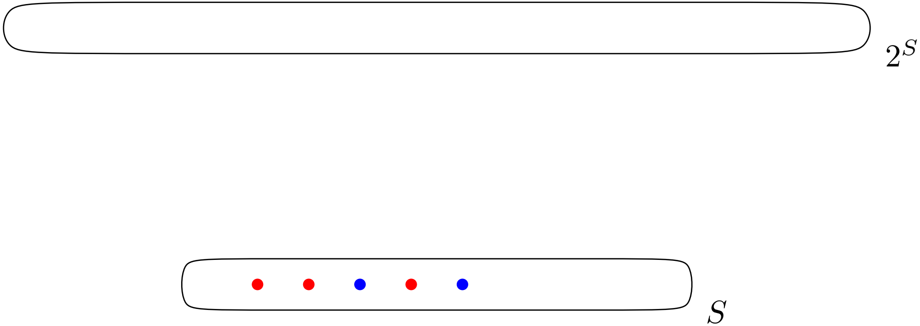
Flip-Breakability $\Rightarrow$ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



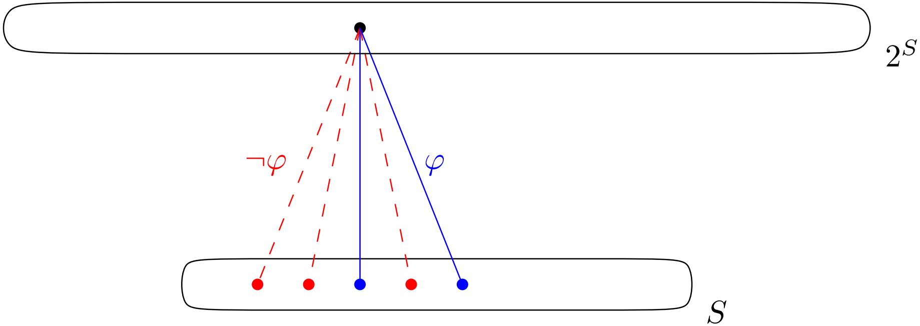
Flip-Breakability $\Rightarrow$ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



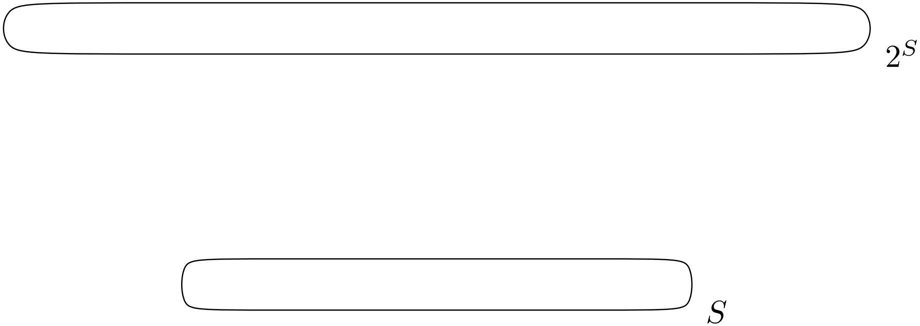
Flip-Breakability $\Rightarrow$ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



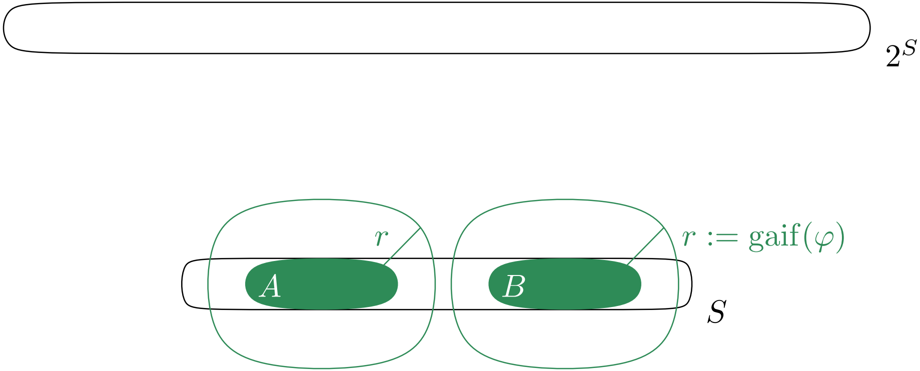
Flip-Breakability ⇒ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



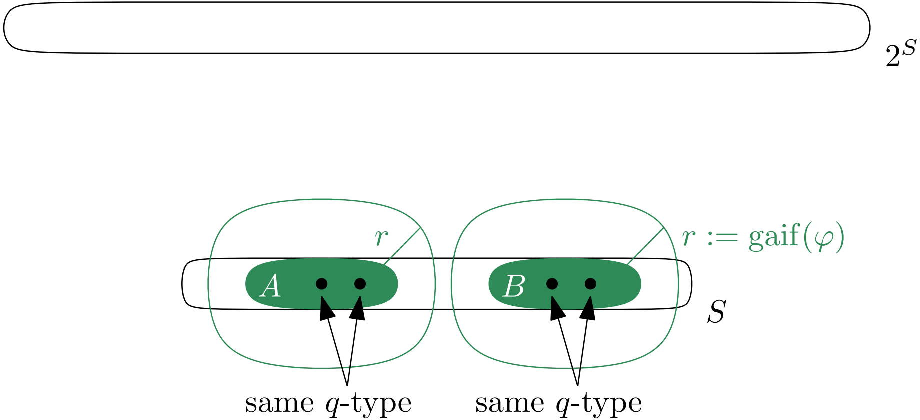
Flip-Breakability $\Rightarrow$ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



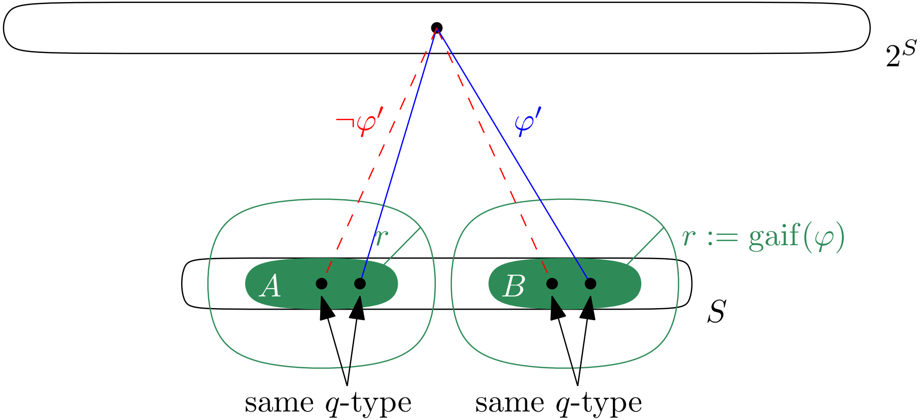
Flip-Breakability $\Rightarrow$ Monadic Dependence

Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



Flip-Breakability $\Rightarrow$ Monadic Dependence

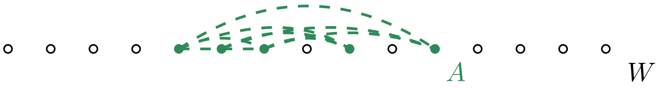
Assume towards a contradiction a class $\mathcal{C}$ is not monadically dependent but flip-breakable.



Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 1$

We prove flip-flatness by induction on r. For $r = 1$ we use Ramsey's theorem.

Case 1: W contains a large independent set.

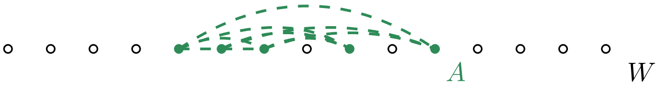


$\rightarrow$ A is distance-1 independent without performing any flips.

Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 1$

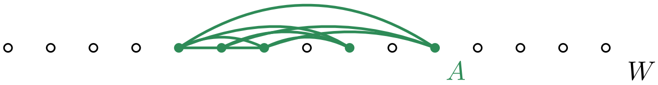
We prove flip-flatness by induction on r. For $r = 1$ we use Ramsey's theorem.

Case 1: W contains a large independent set.



$\rightarrow$ A is distance-1 independent without performing any flips.

Case 2: W contains a large clique.

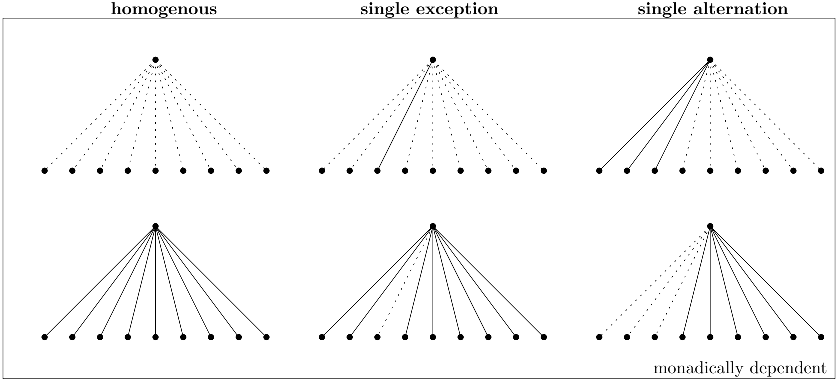


$\rightarrow$ flip (A, A). This is the same as complementing the edges in A.

Monadic Stability ⇒ Flip-Flatness: Indiscernibles

Every long sequence of vertices contains a still long subsequence that is *indiscernible*.

In a monadically dependent class every vertex is connected to an indiscernible sequence in one of the following patterns:

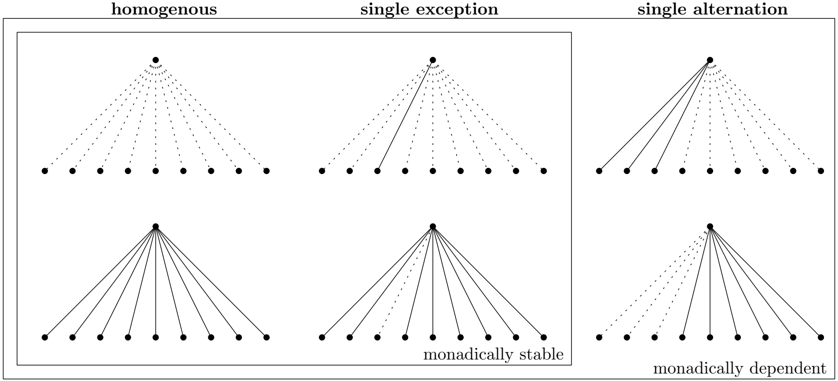


[Blumensath, 2011], [Dreier, Mählmann, Toruńczyk, Siebertz, 2023]

Monadic Stability ⇒ Flip-Flatness: Indiscernibles

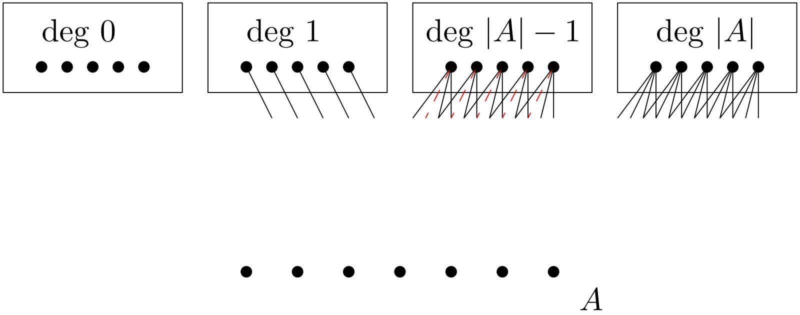
Every long sequence of vertices contains a still long subsequence that is *indiscernible*.

In a monadically dependent class every vertex is connected to an indiscernible sequence in one of the following patterns:

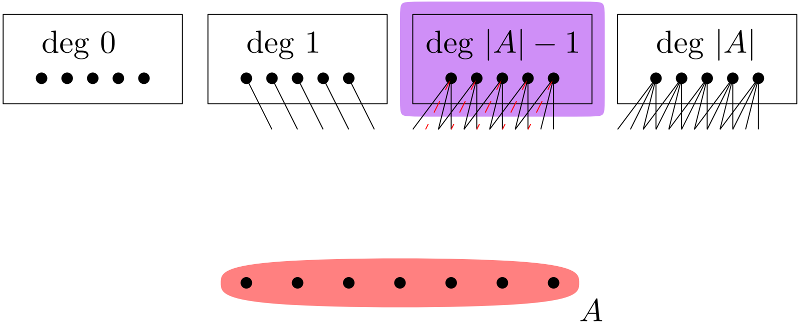


[Blumensath, 2011], [Dreier, Mählmann, Toruńczyk, Siebertz, 2023]

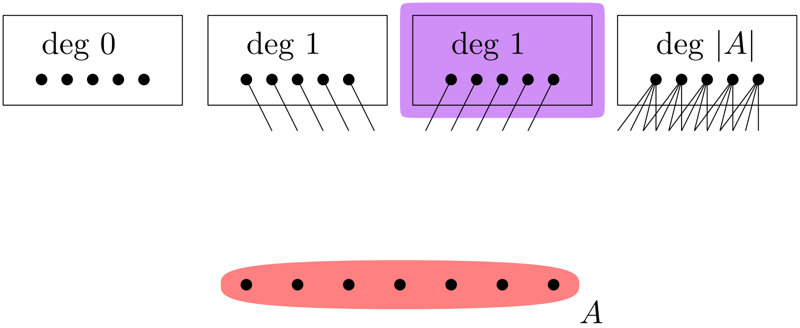
Monadic Stability ⇒ Flip-Flatness: $r = 2$



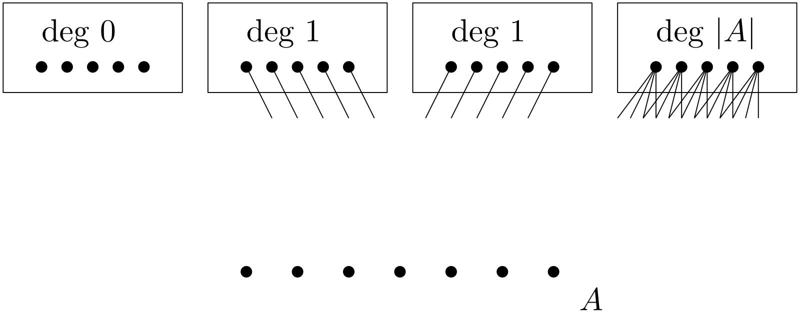
Monadic Stability ⇒ Flip-Flatness: $r = 2$



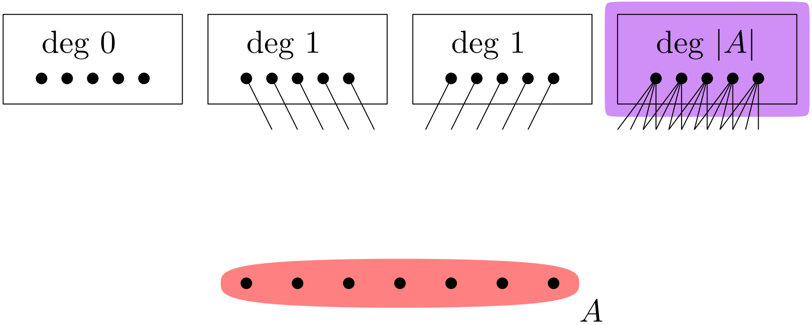
Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



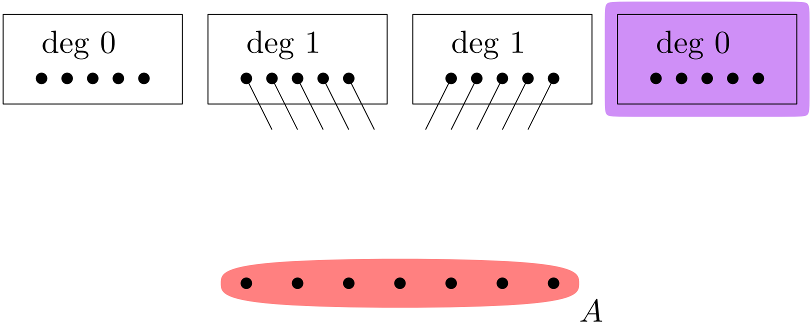
Monadic Stability ⇒ Flip-Flatness: $r = 2$



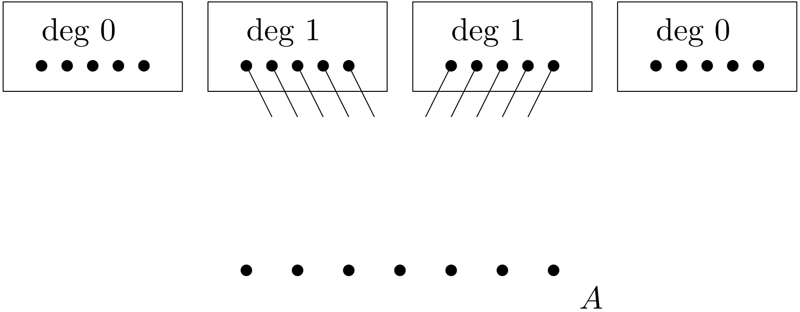
Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



Monadic Stability ⇒ Flip-Flatness: $r = 2$

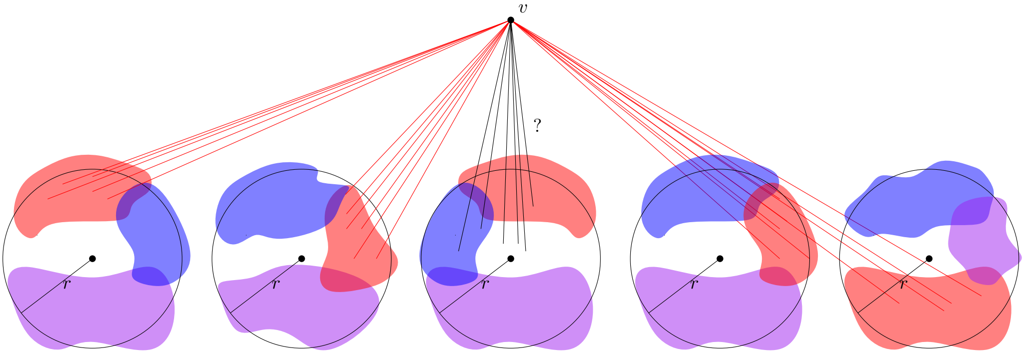


Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



Monadic Stability ⇒ Flip-Flatness: $r \geq 3$

If $\mathcal{C}$ is monadically stable, then every large sequence of disjoint r-balls contains a large subsequence that can be colored by a bounded number of colors such that the neighborhood of every vertex is described by a single colors as follows:



Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability $\Rightarrow$ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability $\Rightarrow$ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability $\Rightarrow$ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability $\Rightarrow$ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability ⇒ Flipper Wins: Proof Idea

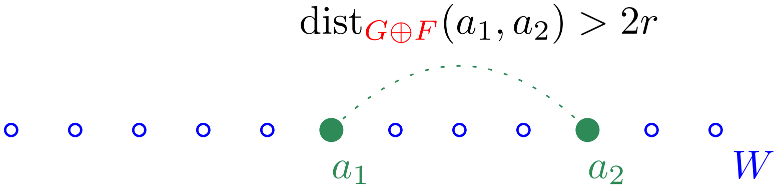
Let $W = w_1, w_2, w_3, \dots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

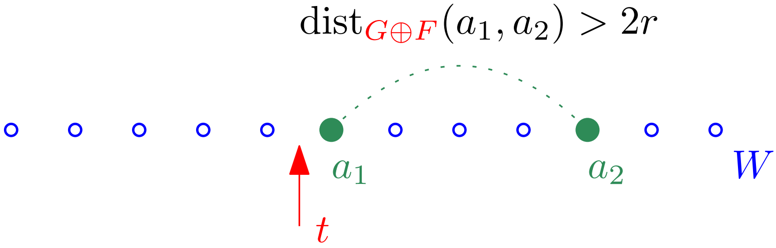
If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.



Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

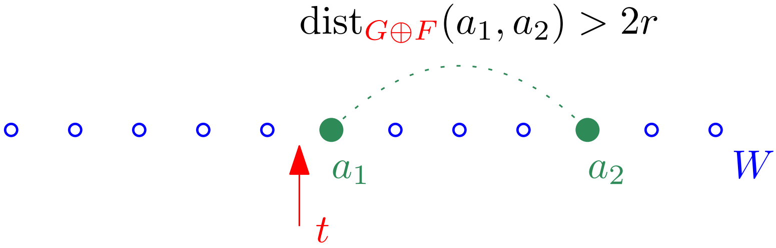
If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.



Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.

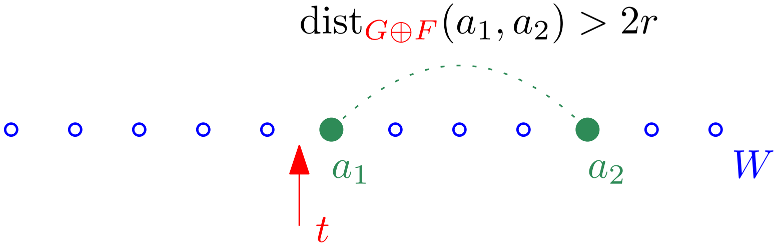


If Flipper had played the flip F at time t then only one of a_1 and a_2 could have survived in the graph.

Mon. Stability ⇒ Flipper Wins: Proof Idea

Let $W = w_1, w_2, w_3, \ldots$ be the vertices played by Localizer.

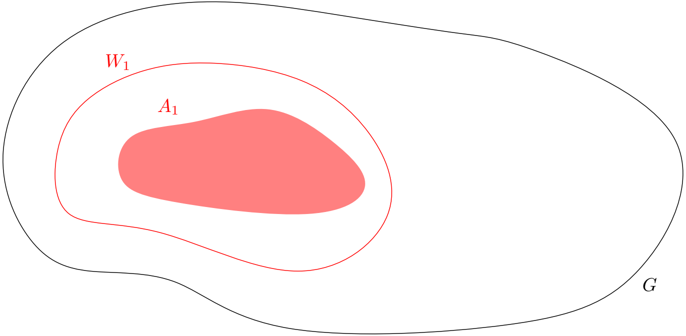
If the game continues long enough, we can apply flip-flatness to find a set $A \subseteq W$ which is $2r$-independent after applying constantly many flips F.



If Flipper had played the flip F at time t then only one of a_1 and a_2 could have survived in the graph.

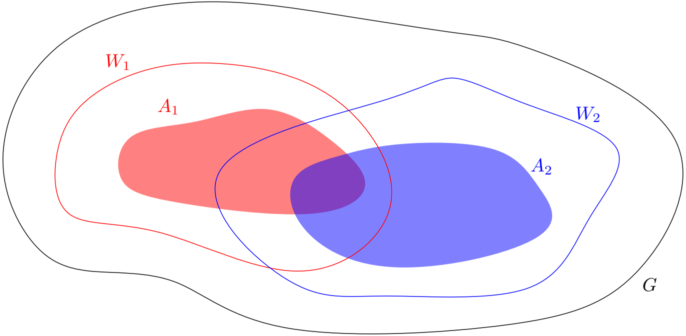
Problem: Flipper does not know W at time t.

Mon. Stability ⇒ Flipper Wins: Predictable Flip-Flatness



$\mathrm{ff}(W_1) = (A_1, F_1)$

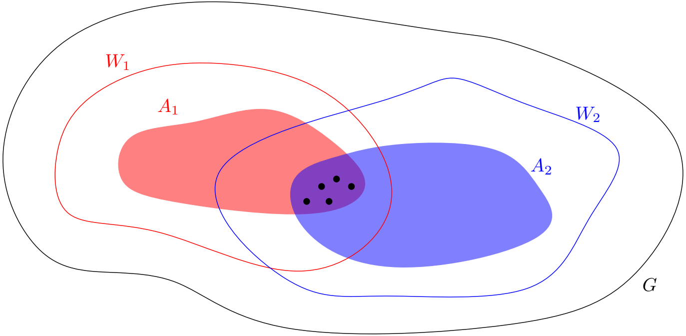
Mon. Stability ⇒ Flipper Wins: Predictable Flip-Flatness



$\mathrm{ff}(W_1) = (A_1, F_1)$

$\mathrm{ff}(W_2) = (A_2, F_2)$

Mon. Stability ⇒ Flipper Wins: Predictable Flip-Flatness



$\mathrm{ff}(W_1) = (A_1, F_1)$

$\mathrm{ff}(W_2) = (A_2, F_2)$

$|A_1 \cap A_2| \geq 5 \quad \Rightarrow \quad F_1 = F_2$

$F_1 = F_2$ are computable from a five-element subset of $A_1 \cap A_2$ in time $\mathcal{O}(n^2)$.

Mon. Stability $\Rightarrow$ Flipper Wins: Flippers Winning Strategy

For every 5 element subset P of Localizers previous moves:

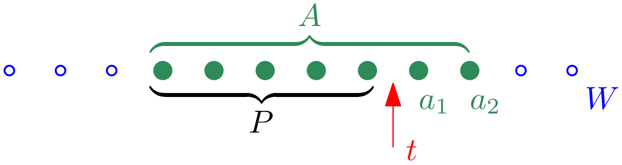
1. apply the flips $\mathrm{predict}(P)$ for radius $2r$
2. let Localizer localize to an r-ball
3. undo $\mathrm{predict}(P)$

Mon. Stability ⇒ Flipper Wins: Flippers Winning Strategy

For every 5 element subset P of Localizers previous moves:

1. apply the flips $\text{predict}(P)$ for radius $2r$
2. let Localizer localize to an r-ball
3. undo $\text{predict}(P)$

Assume Localizer can play enough rounds to apply size 7 flip-flatness

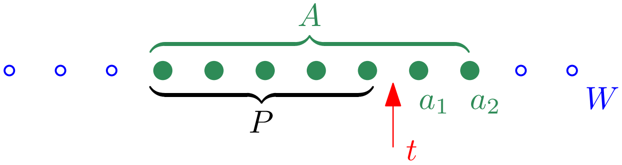


Mon. Stability ⇒ Flipper Wins: Flippers Winning Strategy

For every 5 element subset P of Localizers previous moves:

1. apply the flips $\mathrm{predict}(P)$ for radius $2r$
2. let Localizer localize to an r-ball
3. undo $\mathrm{predict}(P)$

Assume Localizer can play enough rounds to apply size 7 flip-flatness



At time t, P was considered as a subset of Localizers previous moves.

A was flipped $2r$-independent and only one of a_1, a_2 survived. Contradiction!

Model Checking: Idea

Goal: Decide whether $G \models \varphi$.

Model Checking: Idea

Goal: Decide whether $G \models \varphi$.

Idea: Recursion that works by induction on the length ℓ of the Flipper game.

- For every monadically stable class the recursion depth will be bounded.
- For $\ell = 1$ we have $|V(G)| = 1$ and can brute force.

Model Checking: Idea

Goal: Decide whether $G \models \varphi$.

Idea: Recursion that works by induction on the length ℓ of the Flipper game.

- For every monadically stable class the recursion depth will be bounded.
- For $\ell = 1$ we have $|V(G)| = 1$ and can brute force.

We make one round of progress by flipping and localizing.

Model Checking: Idea

Goal: Decide whether $G \models \varphi$.

Idea: Recursion that works by induction on the length ℓ of the Flipper game.

- For every monadically stable class the recursion depth will be bounded.
- For $\ell = 1$ we have $|V(G)| = 1$ and can brute force.

We make one round of progress by flipping and localizing.

Flipping is easy:

- Compute a progressing flip F using Flippers winning strategy
- Rewrite φ and color G such that $G \models \varphi \iff G^+ \oplus F \models \hat{\varphi}$.

Model Checking: Idea

Goal: Decide whether $G \models \varphi$.

Idea: Recursion that works by induction on the length ℓ of the Flipper game.

- For every monadically stable class the recursion depth will be bounded.
- For $\ell = 1$ we have $|V(G)| = 1$ and can brute force.

We make one round of progress by flipping and localizing.

Flipping is easy:

- Compute a progressing flip F using Flippers winning strategy
- Rewrite φ and color G such that $G \models \varphi \iff G^+ \oplus F \models \hat{\varphi}$.

How do we localize? What radius r do we play the Flipper game with?

Model Checking: Guarded Formulas

ψ is $\mathcal{U}$-guarded, if each quantifier is of the form $\exists x \in U$ or $\forall x \in U$ for some $U \in \mathcal{U}$.

Model Checking: Guarded Formulas

ψ is $\mathcal{U}$-guarded, if each quantifier is of the form $\exists x \in U$ or $\forall x \in U$ for some $U \in \mathcal{U}$.

Observation

For every graph G and $\{U_1, \ldots, U_t\}$-guarded formula ψ we have

$$G \models \psi \iff G[U_1 \cup \ldots \cup U_t] \models \psi.$$

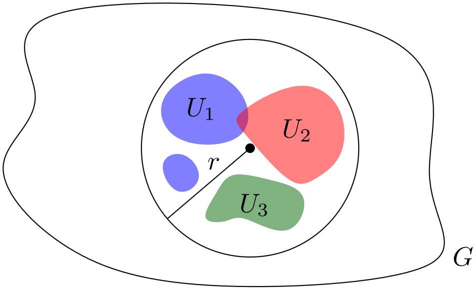
Model Checking: Guarded Formulas

ψ is $\mathcal{U}$-guarded, if each quantifier is of the form $\exists x \in U$ or $\forall x \in U$ for some $U \in \mathcal{U}$.

Observation

For every graph G and $\{U_1, \dots, U_t\}$-guarded formula ψ we have

$$G \models \psi \iff G[U_1 \cup \ldots \cup U_t] \models \psi.$$



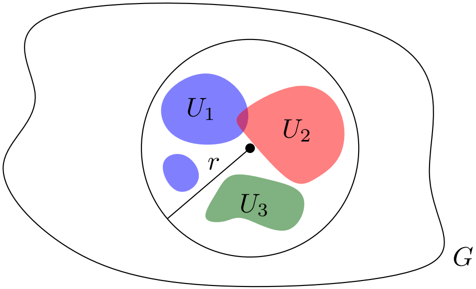
Model Checking: Guarded Formulas

ψ is $\mathcal{U}$-guarded, if each quantifier is of the form $\exists x \in U$ or $\forall x \in U$ for some $U \in \mathcal{U}$.

Observation

For every graph G and $\{U_1, \ldots, U_t\}$-guarded formula ψ we have

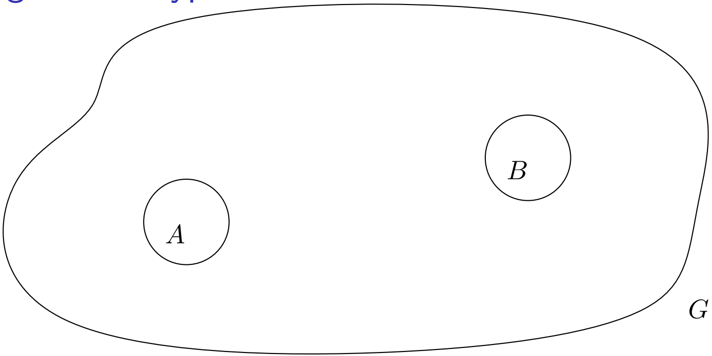
$$G \models \psi \iff G[U_1 \cup \ldots \cup U_t] \models \psi.$$



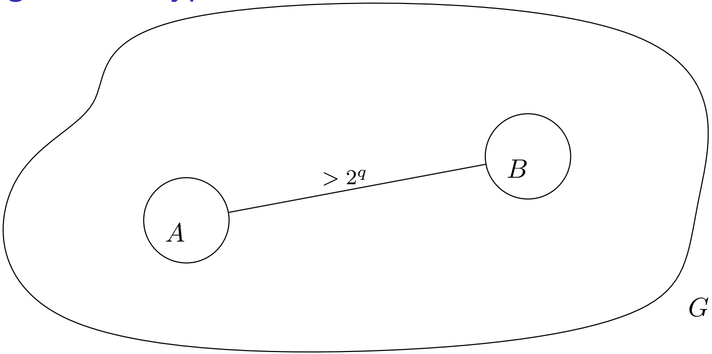
Goal: efficiently compute ψ s.t.

1. ψ is equivalent to φ on G.
2. ψ is a BC of formulas, each guarded by a family of bounded radius in G.

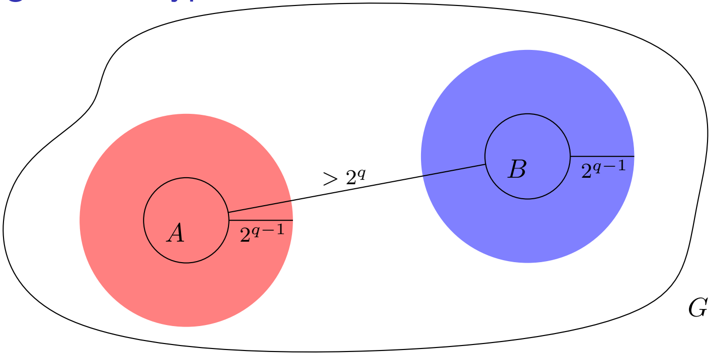
Model Checking: Local Types



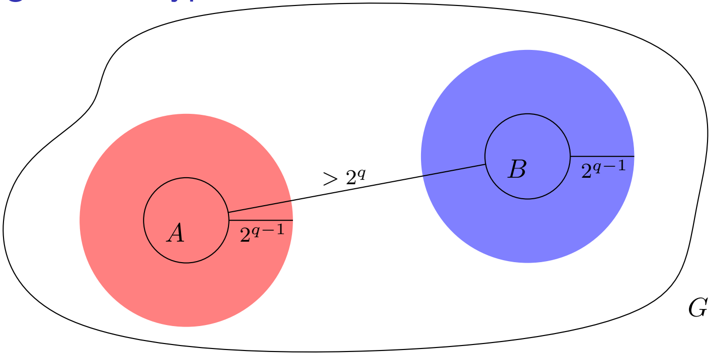
Model Checking: Local Types



Model Checking: Local Types

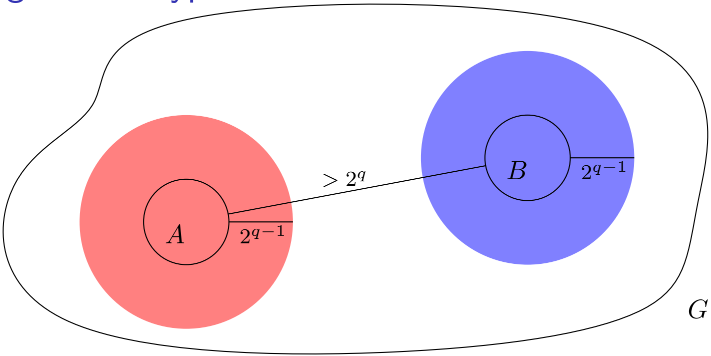


Model Checking: Local Types



Assume $\text{tp}_q(\color{red}{\bullet}) = \text{tp}_q(\color{blue}{\bullet})$. $\text{tp}_q(G) := \{\psi : \psi \text{ has quantifier rank } \leq q \text{ and } G \models \psi\}$

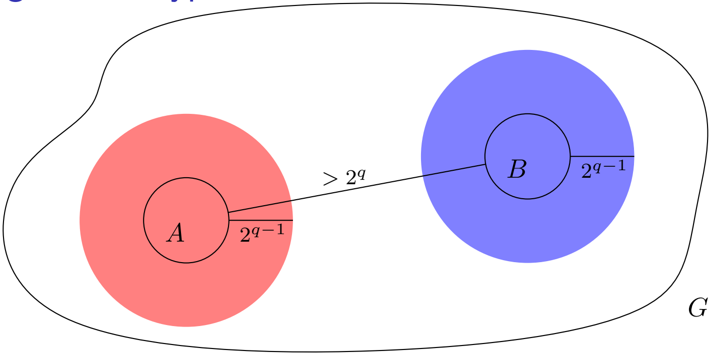
Model Checking: Local Types



Assume $\text{tp}_q(\textcolor{red}{\bullet}) = \text{tp}_q(\textcolor{blue}{\bullet})$. $\text{tp}_q(G) := \{\psi : \psi \text{ has quantifier rank} \leq q \text{ and } G \models \psi\}$

Let $\psi(x)$ be a formula of quantifier rank $q - 1$.

Model Checking: Local Types

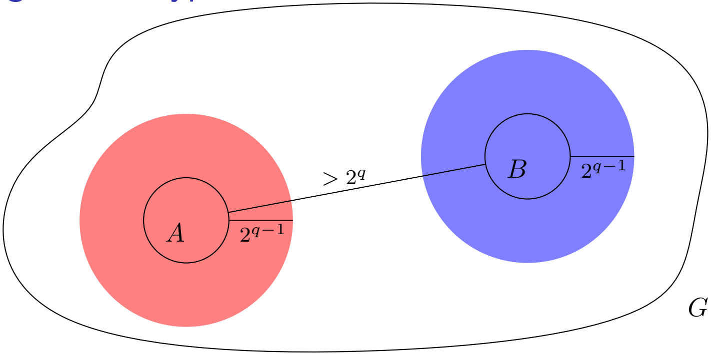


Assume $\mathrm{tp}_q(\textcolor{red}{\bullet}) = \mathrm{tp}_q(\textcolor{blue}{\bullet})$. $\quad$ $\mathrm{tp}_q(G) := \{\psi : \psi \text{ has quantifier rank } \leq q \text{ and } G \models \psi\}$

Let $\psi(x)$ be a formula of quantifier rank $q - 1$.

We have: $G \models \exists x \in A\ \psi(x) \quad \Leftrightarrow \quad G \models \exists x \in B\ \psi(x)$.

Model Checking: Local Types



Assume $\text{tp}_q(\bullet) = \text{tp}_q(\bullet)$. $\text{tp}_q(G) := \{\psi : \psi$ has quantifier rank $\leq q$ and $G \models \psi\}$

Let $\psi(x)$ be a formula of quantifier rank $q - 1$.

We have: $G \models \exists x \in A \ \psi(x) \quad \Leftrightarrow \quad G \models \exists x \in B \ \psi(x)$.

The proof uses a local variant of Ehrenfeucht-Fraïssé games.

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S\ \psi(x).$$

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x \ \psi(x) \quad \Longleftrightarrow \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S \ \psi(x).$$

Every set S is local, but $|\mathcal{S}|$ depends on $|V(G)|$!

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x\ \psi(x) \quad \Longleftrightarrow \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S\ \psi(x).$$

Every set S is local, but $|\mathcal{S}|$ depends on $|V(G)|$!

Idea: Let $\mathcal{S}^\star \subseteq \mathcal{S}$ contain exactly one 2^q-neighborhood for every possible q-type.

By the Local Type Theorem: $G \models \exists x\ \psi(x) \quad \Longleftrightarrow \quad G \models \bigvee_{S \in \mathcal{S}^\star} \exists x \in S\ \psi(x).$

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S\ \psi(x).$$

Every set S is local, but $|\mathcal{S}|$ depends on $|V(G)|$!

Idea: Let $\mathcal{S}^\star \subseteq \mathcal{S}$ contain exactly one 2^q-neighborhood for every possible q-type.

By the Local Type Theorem: $G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}^\star} \exists x \in S\ \psi(x).$

$|\mathcal{S}^\star|$ depends only on q ✓

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S\ \psi(x).$$

Every set S is local, but $|\mathcal{S}|$ depends on $|V(G)|$!

Idea: Let $\mathcal{S}^\star \subseteq \mathcal{S}$ contain exactly one 2^q-neighborhood for every possible q-type.

By the Local Type Theorem: $G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}^\star} \exists x \in S\ \psi(x).$

$|\mathcal{S}^\star|$ depends only on q ✓

When computing $\mathrm{tp}_q(G[S])$, we make progress in the radius-2^q Flipper game ✓

Model Checking: Localizing a Single Quantifier

Let $\mathcal{S} = \{N_{2^q}[v] : v \in V(G)\}$ be the set of 2^q-neighborhoods in G. We have

$$G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}} \exists x \in S\ \psi(x).$$

Every set S is local, but $|\mathcal{S}|$ depends on $|V(G)|$!

Idea: Let $\mathcal{S}^\star \subseteq \mathcal{S}$ contain exactly one 2^q-neighborhood for every possible q-type.

By the Local Type Theorem: $G \models \exists x\ \psi(x) \quad \iff \quad G \models \bigvee_{S \in \mathcal{S}^\star} \exists x \in S\ \psi(x).$

$|\mathcal{S}^\star|$ depends only on q ✓

When computing $\mathrm{tp}_q(G[S])$, we make progress in the radius-2^q Flipper game ✓

For multiple quantifiers: extend to parameters and argue by induction ✓

Model Checking: Recursion Tree

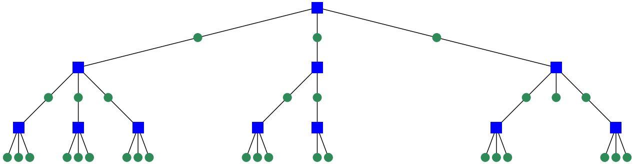
We can now play the Flipper game for radius 2^q:

1. Flip by rewriting φ and coloring G.
2. Localize by computing the q-type of every 2^q-neighborhood.

Model Checking: Recursion Tree

We can now play the Flipper game for radius 2^q:

1. Flip by rewriting φ and coloring G.
2. Localize by computing the q-type of every 2^q-neighborhood.

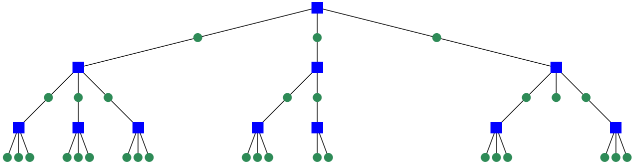


By monadic stability the depth of the recursion tree is bounded by $f(q)$.

Model Checking: Recursion Tree

We can now play the Flipper game for radius 2^q:

1. Flip by rewriting φ and coloring G.
2. Localize by computing the q-type of every 2^q-neighborhood.

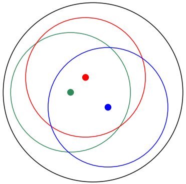


By monadic stability the depth of the recursion tree is bounded by $f(q)$.

However the branching degree is n. This gives an $\mathcal{O}(n^{f(q)})$ algorithm.

This is worse than the naive $\mathcal{O}(n^q)$ algorithm!

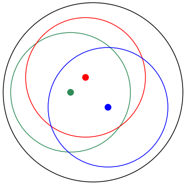
Model Checking: Neighborhood Covers



Recursing into each 2^q-neighborhood is too expensive!

Idea: group neighborhoods that are close to each other into clusters.

Model Checking: Neighborhood Covers



Recursing into each 2^q-neighborhood is too expensive!

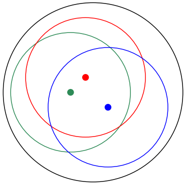
Idea: group neighborhoods that are close to each other into clusters.

Definition

A family of sets $\mathcal{X}$ is a *neighborhood cover* with radius r, spread s, and degree d if

- each r-neighborhood of G is fully contained in one cluster $X \in \mathcal{X}$,
- each cluster is contained in an s-neighborhood of G,
- each vertex appears in at most d clusters.

Model Checking: Neighborhood Covers



Recursing into each 2^q-neighborhood is too expensive!

Idea: group neighborhoods that are close to each other into clusters.

Definition

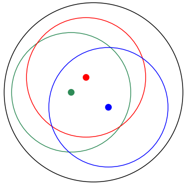
A family of sets $\mathcal{X}$ is a *neighborhood cover* with radius r, spread s, and degree d if

- each r-neighborhood of G is fully contained in one cluster $X \in \mathcal{X}$,
- each cluster is contained in an s-neighborhood of G,
- each vertex appears in at most d clusters.

A class admits sparse neighborhood covers if we can set $d = g(r, \varepsilon) \cdot n^{\varepsilon}$ for every $\varepsilon > 0$.

Model Checking: Neighborhood Covers

Recursing into each 2^q-neighborhood is too expensive!



Idea: group neighborhoods that are close to each other into clusters.

Definition

A family of sets $\mathcal{X}$ is a *neighborhood cover* with radius r, spread s, and degree d if

- each r-neighborhood of G is fully contained in one cluster $X \in \mathcal{X}$,
- each cluster is contained in an s-neighborhood of G,
- each vertex appears in at most d clusters.

A class admits sparse neighborhood covers if we can set $d = g(r, \varepsilon) \cdot n^{\varepsilon}$ for every $\varepsilon > 0$.

The size of the clusters of a sparse neighborhood cover sum up to $g(r, \varepsilon) \cdot n^{1+\varepsilon}$.

Resulting size of the recursion tree: $n^{((1+\varepsilon)^{f(q)})}$; by choosing ε small enough: $n^{1+\varepsilon'}$.

Model Checking: Summary

Theorem [Dreier, Mählmann, Siebertz, 2023]

Every monadically stable class, that admits sparse neighborhood covers, admits FO model checking in time $f(\varphi) \cdot |V(G)|^{11}$.

Theorem [Dreier, Mählmann, Siebertz, 2023]

Every structurally nowhere dense class admits sparse neighborhood covers.

Model Checking: Summary

Theorem [Dreier, Mählmann, Siebertz, 2023]

Every monadically stable class, that admits sparse neighborhood covers, admits FO model checking in time $f(\varphi) \cdot |V(G)|^{11}$.

Theorem [Dreier, Mählmann, Siebertz, 2023]

Every structurally nowhere dense class admits sparse neighborhood covers.

Theorem [Dreier, Eleftheriadis, Mählmann, McCarty, Pilipczuk, Toruńczyk, 2023]

Every monadically stable class admits sparse neighborhood covers.

Theorem

Every monadically stable class admits FO model checking in time $f(\varphi, \varepsilon) \cdot |V(G)|^{6+\varepsilon}$.

Stability and Dependence in Model Theory

On a class $\mathcal{C}$, a formula $\varphi(\bar{x}, \bar{y})$ has

- the *order property* if for every $k \in \mathbb{N}$ there are $G \in \mathcal{C}$ and two sequences $(\bar{a}_i)_{i\in[k]}$, $(\bar{b}_j)_{j\in[k]}$ of tuples in G, such that for all $i, j \in [k]$: $G \models \varphi(\bar{a}_i, \bar{b}_j) \Leftrightarrow i \le j$.
- the *independence property* if for every $k \in \mathbb{N}$ there are $G \in \mathcal{C}$, a size k set $A \subseteq V(G)^{|\bar{x}|}$ and a sequence $(\bar{b}_J)_{J\subseteq A}$ of tuples in G such that for all $\bar{a} \in A, J \subseteq A$

$$G \models \varphi(\bar{a}, \bar{b}_J) \quad \Leftrightarrow \quad \bar{a} \in J.$$

A graph class is *stable* if it does not have the order property.
It is *monadically stable* if the class of colored graphs from $\mathcal{C}$ is stable.

A graph class is *dependent* if it does not have the independence property.
It is *monadically dependent* if the class of colored graphs from $\mathcal{C}$ is dependent.

Approximation Algorithms

Distance-r dominating set:

- constant factor approximation in bounded expansion classes [Dvořák 2013]
- $O(d \cdot \log(d \cdot OPT))$ approximation of the distance-1 case on graphs with VC dimension $\leq d$ [Brönnimann, Goodrich, 1995]

Distance-r independent set:

- constant factor approximation in bounded expansion classes [Dvořák 2013]
- n^{ε} approximation in nowhere dense classes [Dvořák 2019]
- n^{ε} approximation in bounded twin-width classes [Bergé et al. 2022]