Sparsity Theory for Dense Graphs

Nikolas Mählmann

Sparsity - Graphs and Algorithms 01.02.2024

The Model Checking Problem

Problem: Given a graph G and a formula φ in a logic $\mathcal{L} \in \{FO, MSO\}$, decide $G \models \varphi$.

Examples:

- FO: distance-r red blue independent/dominating set of size k
- MSO: 3-colorability, SAT

The Model Checking Problem

Problem: Given a graph G and a formula φ in a logic $\mathcal{L} \in \{FO, MSO\}$, decide $G \models \varphi$.

Examples:

- FO: distance-r red blue independent/dominating set of size k
- MSO: 3-colorability, SAT

Runtime:

- FO: $\mathcal{O}(n^q)$ (q = quantifier rank; assuming ETH)
- MSO: NP-hard

The Model Checking Problem

Problem: Given a graph G and a formula φ in a logic $\mathcal{L} \in \{FO, MSO\}$, decide $G \models \varphi$.

Examples:

- FO: distance-r red blue independent/dominating set of size k
- MSO: 3-colorability, SAT

Runtime:

- FO: $\mathcal{O}(n^q)$ (q = quantifier rank; assuming ETH)
- MSO: NP-hard

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

Monadic Second-Order Logic

MSO Model Checking

Theorem [Courcelle, 1990]

Every class of bounded treewidth admits MSO model checking in time $f(\varphi) \cdot n$.

MSO Model Checking

Theorem [Courcelle, 1990]

Every class of bounded treewidth admits MSO model checking in time $f(\varphi) \cdot n$.

The class of all cliques has unbounded treewidth but model checking is trivial.

MSO Model Checking

Theorem [Courcelle, 1990]

Every class of bounded treewidth admits MSO model checking in time $f(\varphi) \cdot n$.

The class of all cliques has unbounded treewidth but model checking is trivial.

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)

MSO Model Checking

Theorem [Courcelle, 1990]

Every class of bounded treewidth admits MSO model checking in time $f(\varphi) \cdot n$.

The class of all cliques has unbounded treewidth but model checking is trivial.

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)

Treedepth, treewidth, minors, bounded expansion, nowhere denseness, etc.
are all measures for monotone graph classes.

To handle dense graphs we need complexity measures for hereditary graph classes!

Classes of Bounded Cliquewidth

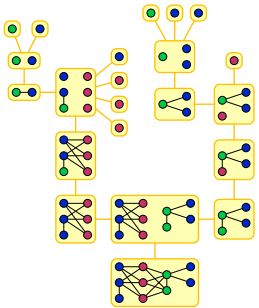


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.

Classes of Bounded Cliquewidth

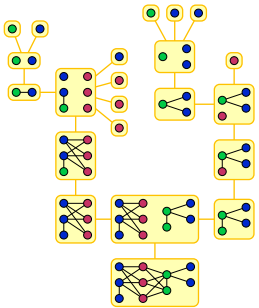


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.

Classes of Bounded Cliquewidth

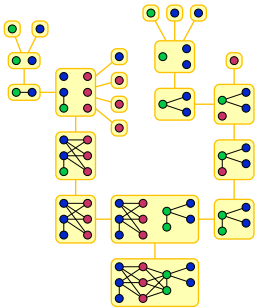


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.
3. Connecting all vertices with label i to all vertices with label j.

Classes of Bounded Cliquewidth

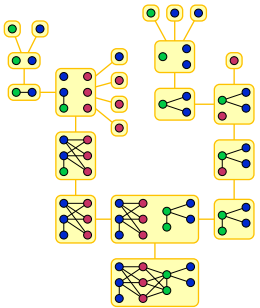


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.
3. Connecting all vertices with label i to all vertices with label j.
4. Relabeling all vertices with label i to label j.

Classes of Bounded Cliquewidth

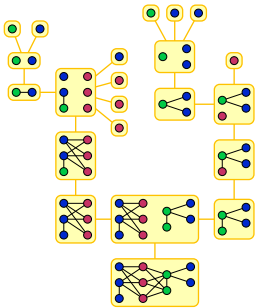


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.
3. Connecting all vertices with label i to all vertices with label j.
4. Relabeling all vertices with label i to label j.

The cliquewidth of a graph is the minimum number of labels needed to construct it.

Classes of Bounded Cliquewidth

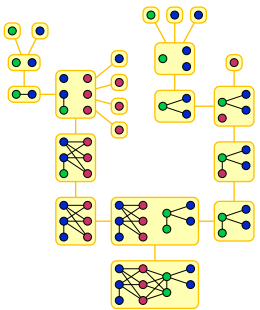


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.
3. Connecting all vertices with label i to all vertices with label j.
4. Relabeling all vertices with label i to label j.

The cliquewidth of a graph is the minimum number of labels needed to construct it.

Examples: Cliques have cliquewidth 1. Half-graphs have cliquewidth ≤ 3.

Classes of Bounded Cliquewidth

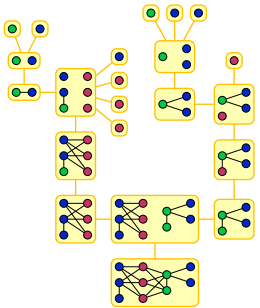


figure by David Eppstein under CC0

Labeled graphs of cliquewidth k are constructed using the following operations:

1. Creating a vertex with label $i \in [k]$.
2. Taking the disjoint union of two labeled graphs.
3. Connecting all vertices with label i to all vertices with label j.
4. Relabeling all vertices with label i to label j.

The cliquewidth of a graph is the minimum number of labels needed to construct it.

Examples: Cliques have cliquewidth 1. Half-graphs have cliquewidth ≤ 3.

Theorem [Courcelle, Makowsky, Rotics, 2000] [Oum, Seymour, 2006]

Every class of bounded cliquewidth admits MSO model checking in time $f(\varphi) \cdot n^3$.

First-Order Logic

Nowhere Dense Classes of Graphs

Definition [Nešetřil, Ossona de Mendez, 2011]

A class $\mathcal{C}$ is *nowhere dense*, if for every r there exists k such $\mathcal{C}$ that does not contain the r-subdivided clique of size k as a subgraph.

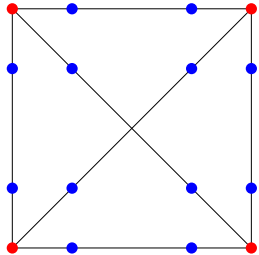


Figure: The 2-subdivided K_4.

Nowhere Dense Classes of Graphs

Definition [Nešetřil, Ossona de Mendez, 2011]

A class $\mathcal{C}$ is *nowhere dense*, if for every r there exists k such $\mathcal{C}$ that does not contain the r-subdivided clique of size k as a subgraph.

Generalizes many notions of sparsity such as: bounded degree, bounded treewidth, planarity, excluding a minor, ...

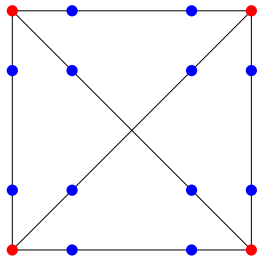


Figure: The 2-subdivided K_4.

Nowhere Dense Classes of Graphs

Definition [Nešetřil, Ossona de Mendez, 2011]

A class $\mathcal{C}$ is *nowhere dense*, if for every r there exists k such $\mathcal{C}$ that does not contain the r-subdivided clique of size k as a subgraph.

Generalizes many notions of sparsity such as: bounded degree, bounded treewidth, planarity, excluding a minor, ...

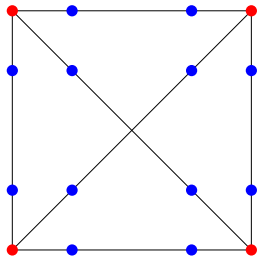


Figure: The 2-subdivided K_4.

Theorem [Grohe, Kreutzer, Siebertz, 2014]

Let $\mathcal{C}$ be a *monotone* class of graphs. If $\mathcal{C}$ is nowhere dense, then FO model checking on $\mathcal{C}$ can be done in time $f(\varphi, \varepsilon) \cdot n^{1+\varepsilon}$ for every $\varepsilon > 0$. Otherwise it is AW[$*$]-hard.

FO Transductions

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

FO Transductions

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

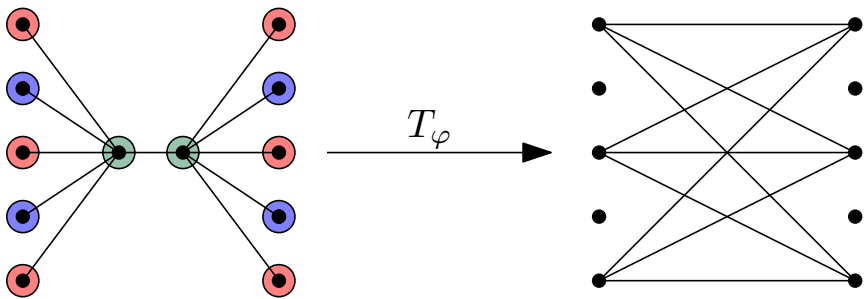
How to produce well behaved hereditary classes from sparse classes?

FO Transductions

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

How to produce well behaved hereditary classes from sparse classes?

Transductions $\hat{=}$ coloring + interpreting + taking an induced subgraph



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

Structural Sparsity and Monadic Stability

Definition

A class $\mathcal{C}$ is *structurally nowhere dense*, if there exists a transduction T and a nowhere dense class $\mathcal{D}$ such that $\mathcal{C} \subseteq T(\mathcal{D})$.

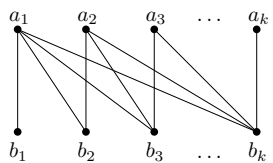
Structural Sparsity and Monadic Stability

Definition

A class $\mathcal{C}$ is *structurally nowhere dense*, if there exists a transduction T and a nowhere dense class $\mathcal{D}$ such that $\mathcal{C} \subseteq T(\mathcal{D})$.

Definition

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



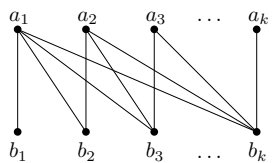
Structural Sparsity and Monadic Stability

Definition

A class $\mathcal{C}$ is *structurally nowhere dense*, if there exists a transduction T and a nowhere dense class $\mathcal{D}$ such that $\mathcal{C} \subseteq T(\mathcal{D})$.

Definition

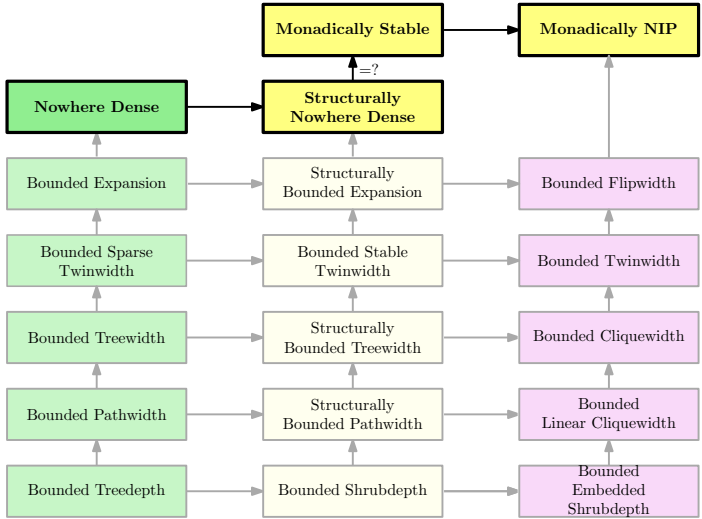
A class is *monadically stable*, if it does not transduce the class of all half graphs.



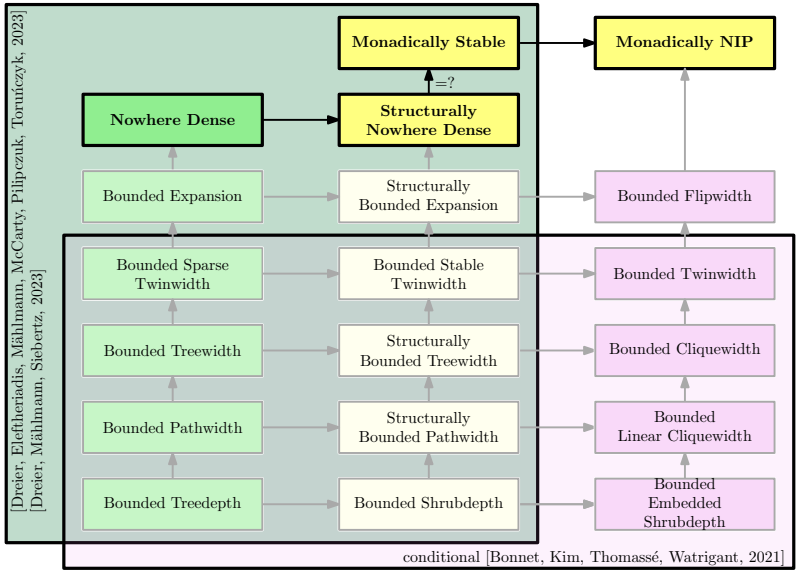
Definition

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

Map of the Universe



Map of the Universe



Tractable Classes

Theorem [Grohe, Kreutzer, Siebertz, 2014]

Let $\mathcal{C}$ be a monotone class of graphs.

$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ is nowhere dense.

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

Let $\mathcal{C}$ be a hereditary and orderless class of graphs.

$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ is monadically stable.

Theorem [Bonnet, Giocanti, Ossona de Mendez, Simon, Thomassé, Toruńczyk, 2022]

Let $\mathcal{C}$ be a hereditary and ordered class of graphs.

$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ has bounded twin-width.

Tractable Classes

Theorem [Grohe, Kreutzer, Siebertz, 2014]

Let $\mathcal{C}$ be a monotone class of graphs.

$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ is **monadically NIP**.

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

Let $\mathcal{C}$ be a hereditary and orderless class of graphs.

$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ is **monadically NIP**.

Theorem [Bonnet, Giocanti, Ossona de Mendez, Simon, Thomassé, Toruńczyk, 2022]

Let $\mathcal{C}$ be a hereditary and ordered class of graphs.

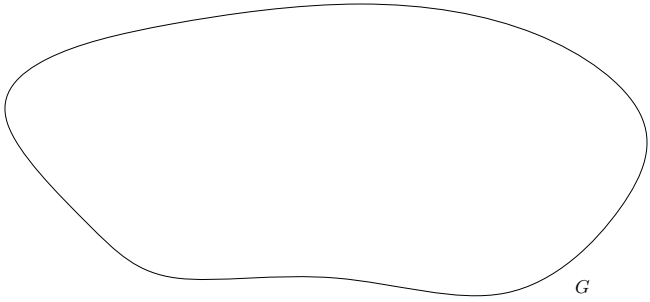
$\mathcal{C}$ admits fpt FO model checking if and only if $\mathcal{C}$ is **monadically NIP**.

Agenda

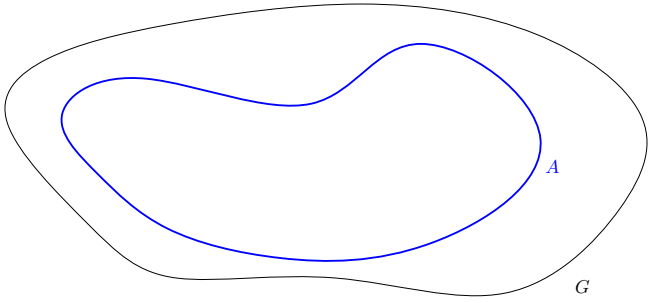
I will present some of our research results:

1. Characterizations of monadically stable and monadically NIP classes.
2. A game for monadically stable graph classes.
3. FO model checking for monadically stable graph classes.

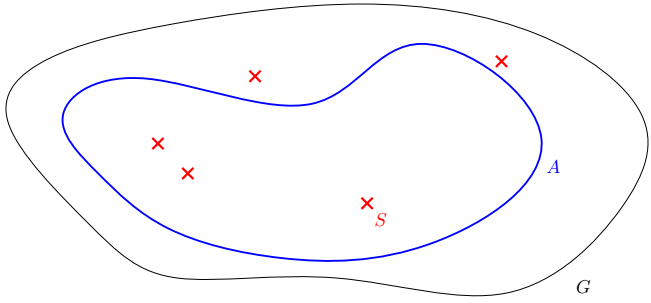
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



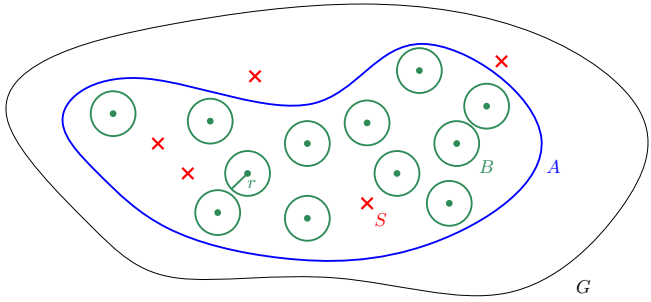
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



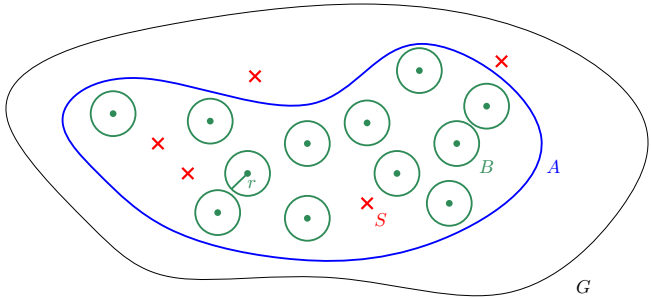
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



Characterizing Nowhere Denseness: Uniform Quasi-Wideness



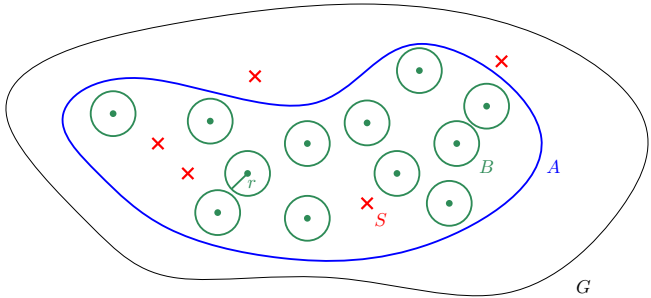
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



Uniform Quasi-Wideness slightly informal)

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

Characterizing Nowhere Denseness: Uniform Quasi-Wideness

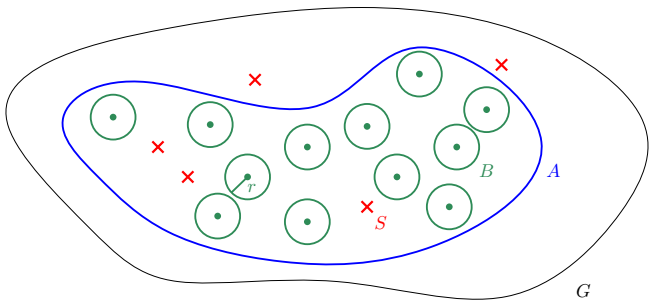


Uniform Quasi-Wideness slightly informal)

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

Formally:
$\forall r\ \exists s_r \in \mathbb{N}, N_r : \mathbb{N} \to \mathbb{N}$ s.t. $\forall G \in \mathcal{C}, A \subseteq V(G)$ with $|A| \geq N_r(m)$
$\exists B \subseteq A, S \subseteq V(G)$ with $|B| \geq m, |S| \leq s_r$ s.t. B is r-independent in $G - S$.

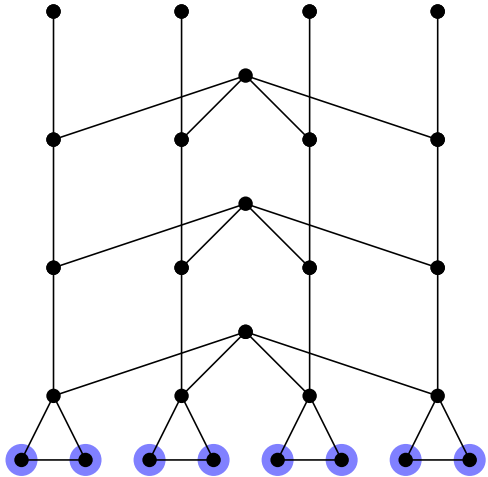
Characterizing Nowhere Denseness: Uniform Quasi-Wideness



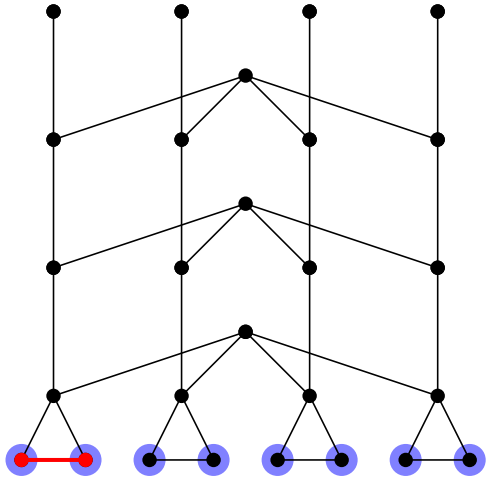
Theorem [Nešetřil, Ossona de Mendez, 2011]

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

Uniform Quasi-Wideness: Example

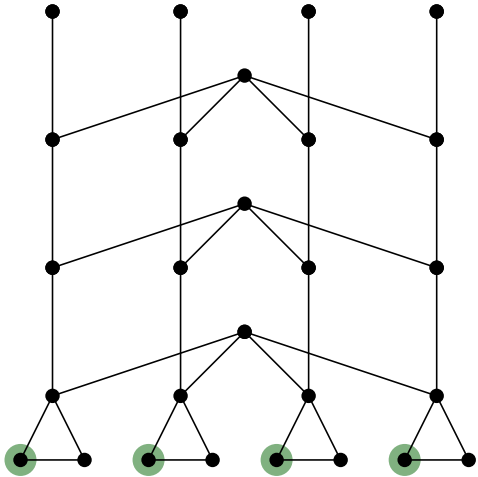


Uniform Quasi-Wideness: Example

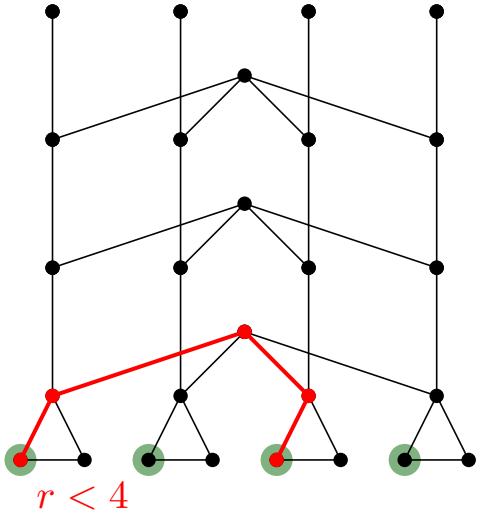


$r < 1$

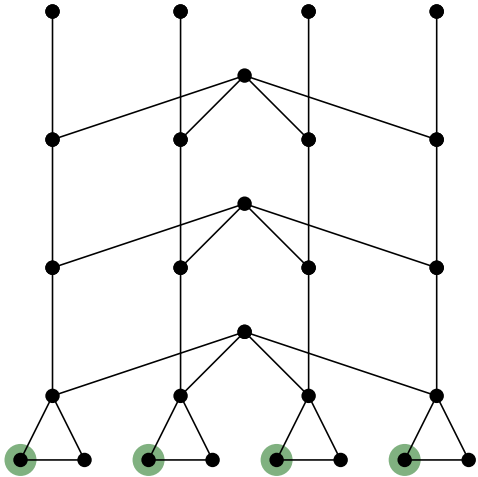
Uniform Quasi-Wideness: Example



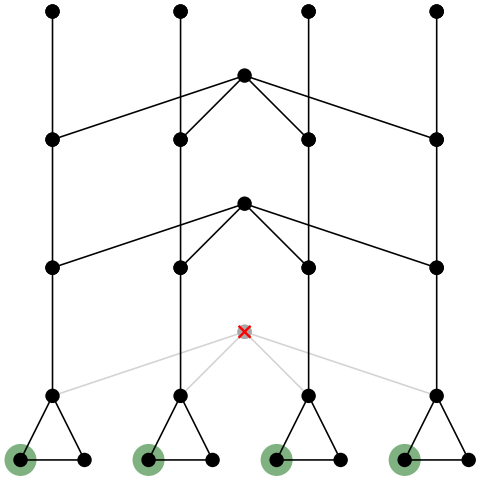
Uniform Quasi-Wideness: Example



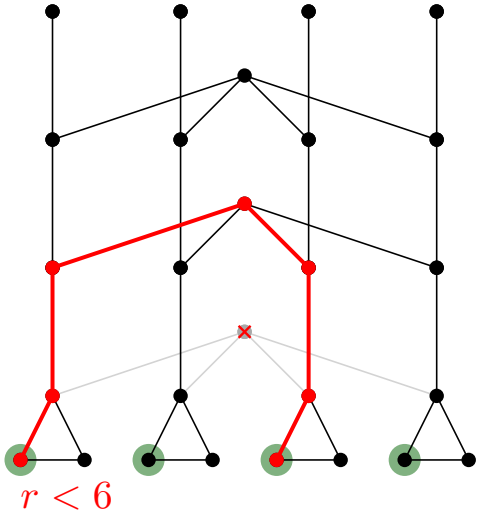
Uniform Quasi-Wideness: Example



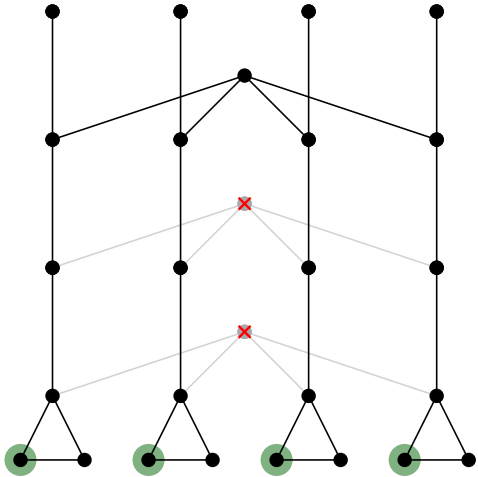
Uniform Quasi-Wideness: Example



Uniform Quasi-Wideness: Example



Uniform Quasi-Wideness: Example



Flips

Theorem [Nešetřil, Ossona de Mendez, 2011]

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

Question: Similar **combinatorial characterizations** for monadic stability/NIP?

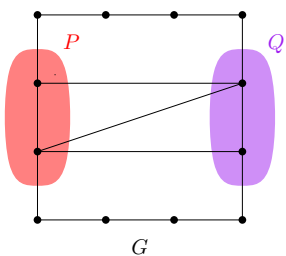
Flips

Theorem [Nešetřil, Ossona de Mendez, 2011]

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

Question: Similar **combinatorial characterizations** for monadic stability/NIP?

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



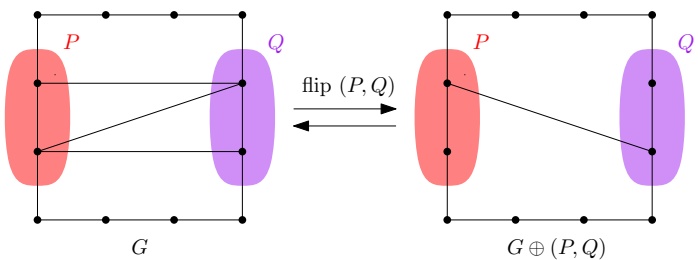
Flips

Theorem [Nešetřil, Ossona de Mendez, 2011]

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

Question: Similar **combinatorial characterizations** for monadic stability/NIP?

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



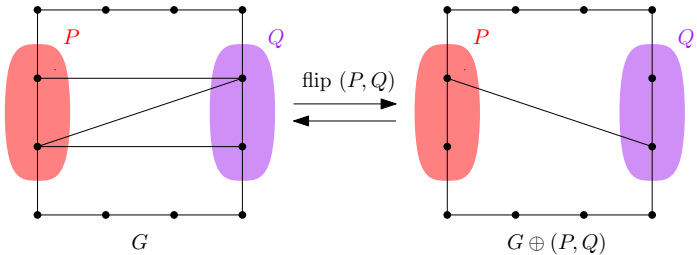
Flips

Theorem [Nešetřil, Ossona de Mendez, 2011]

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

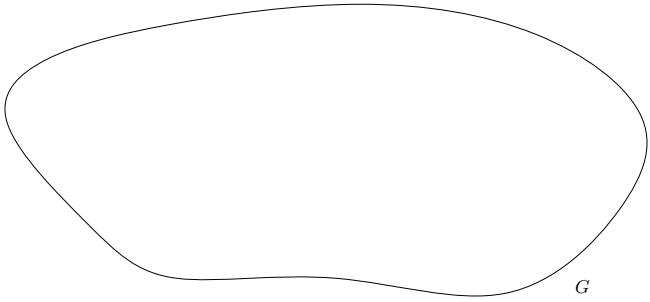
Question: Similar **combinatorial characterizations** for monadic stability/NIP?

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

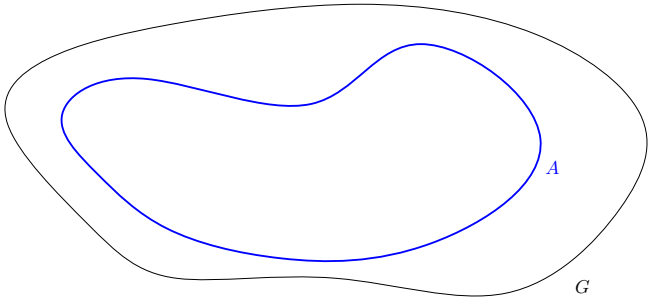


$$G \models E(u,v) \quad \Leftrightarrow \quad G \oplus (P,Q) \models E(u,v) \ \texttt{XOR} \ \big(P(u) \wedge Q(v) \ \vee \ P(v) \wedge Q(u)\big)$$

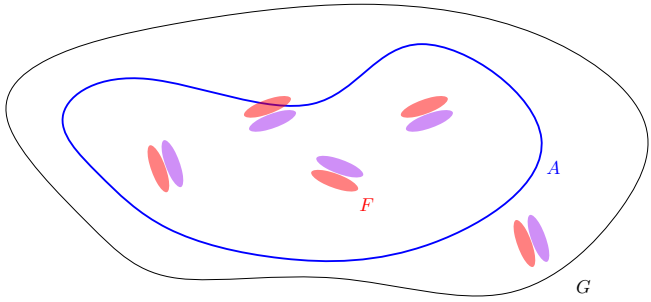
Characterizing Monadic Stability: Flip-Flatness



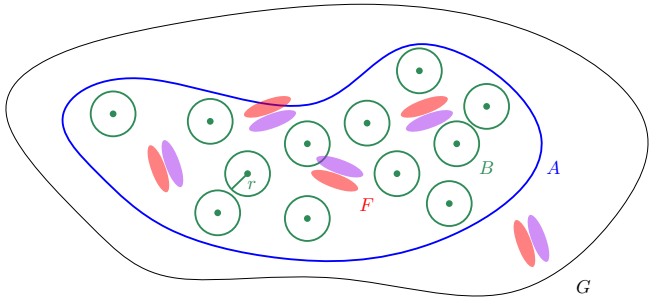
Characterizing Monadic Stability: Flip-Flatness



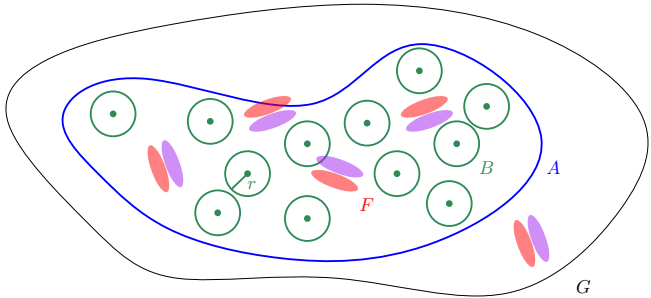
Characterizing Monadic Stability: Flip-Flatness



Characterizing Monadic Stability: Flip-Flatness



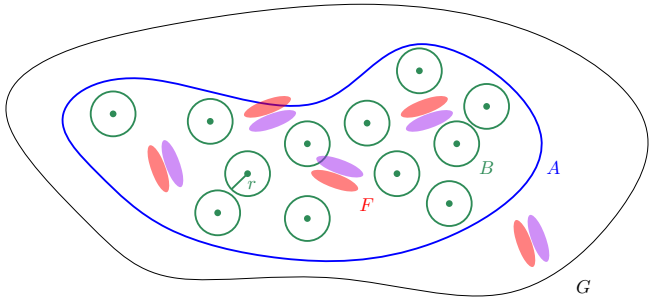
Characterizing Monadic Stability: Flip-Flatness



Flip-Flatness (slightly informal) [Gajarský, Kreutzer]

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

Characterizing Monadic Stability: Flip-Flatness



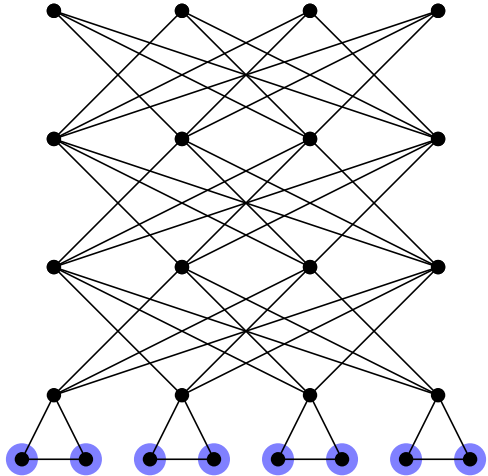
Flip-Flatness (slightly informal) [Gajarský, Kreutzer]

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

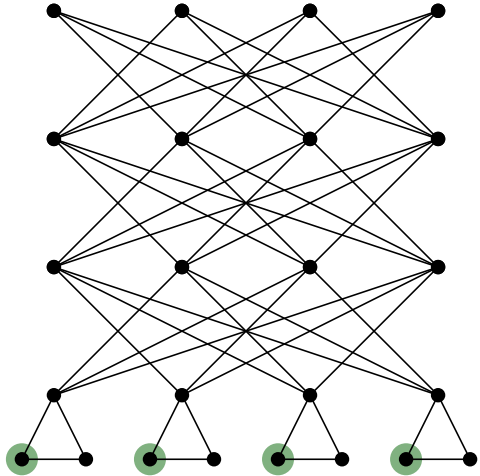
Theorem [Dreier, Mählmann, Siebertz, Toruńczyk, 2022]

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

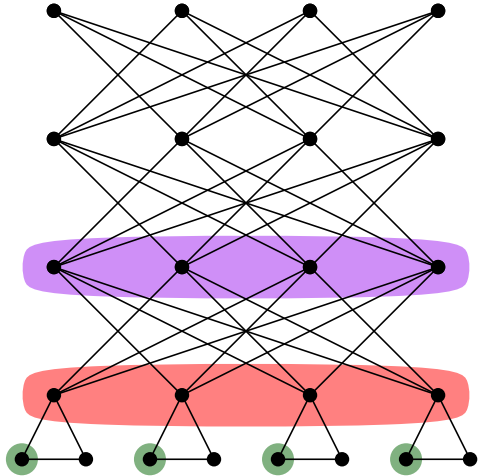
Flip-Flatness: Example



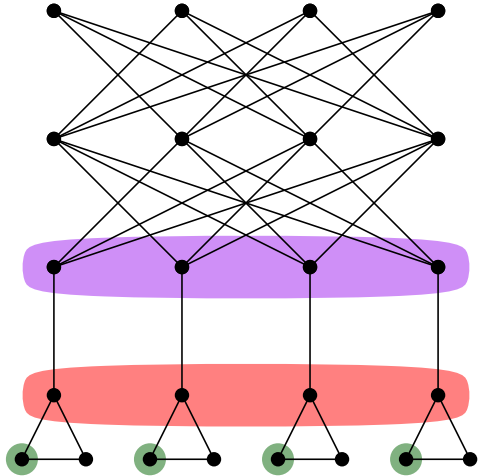
Flip-Flatness: Example



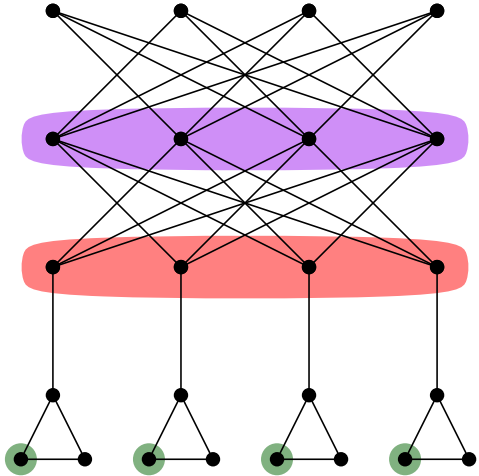
Flip-Flatness: Example



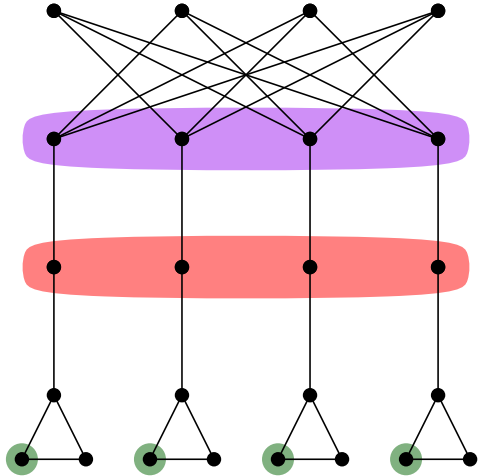
Flip-Flatness: Example



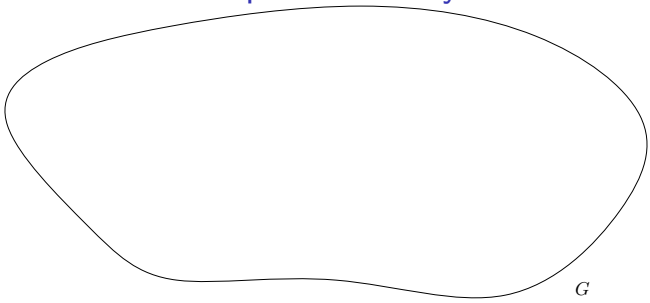
Flip-Flatness: Example



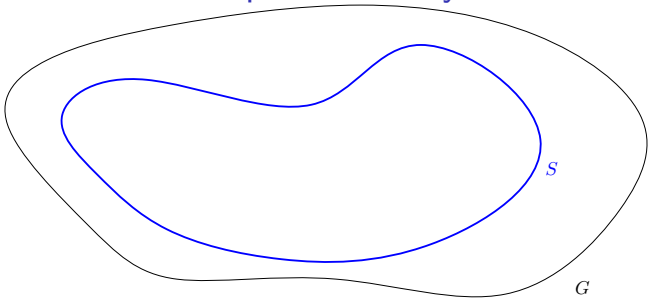
Flip-Flatness: Example



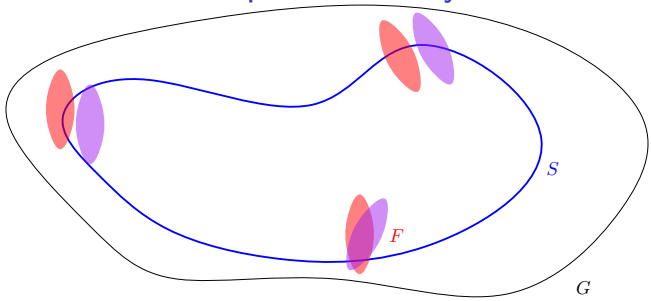
Characterizing Monadic NIP: Flip-Breakability



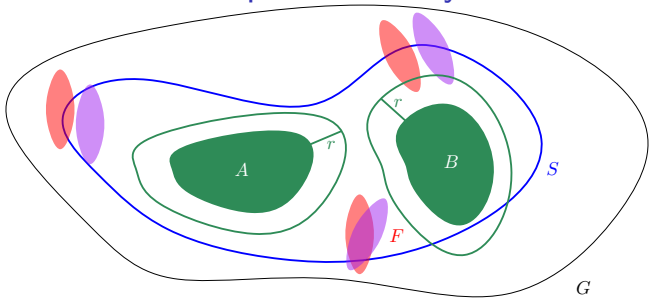
Characterizing Monadic NIP: Flip-Breakability



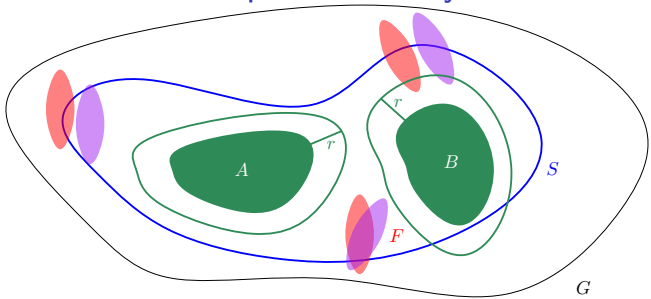
Characterizing Monadic NIP: Flip-Breakability



Characterizing Monadic NIP: Flip-Breakability



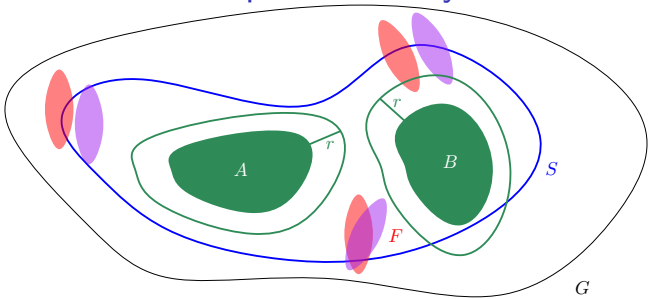
Characterizing Monadic NIP: Flip-Breakability



Flip-Breakability (slightly informal)

A class $\mathcal{C}$ is *flip-breakable* if for every radius r, in every large set S we find two large sets A and B that and a flip F of bounded size such that $N^r_{G \oplus F}(A) \cap N^r_{G \oplus F}(B) = \varnothing$.

Characterizing Monadic NIP: Flip-Breakability



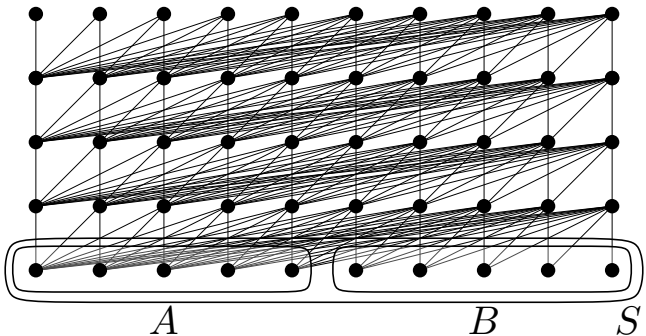
Flip-Breakability (slightly informal)

A class $\mathcal{C}$ is *flip-breakable* if for every radius r, in every large set S we find two large sets A and B that and a flip F of bounded size such that $N^r_{G \oplus F}(A) \cap N^r_{G \oplus F}(B) = \varnothing$.

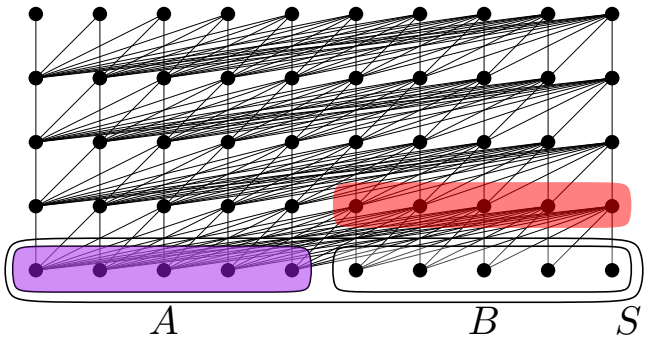
Theorem [Dreier, Mählmann, Toruńczyk]

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

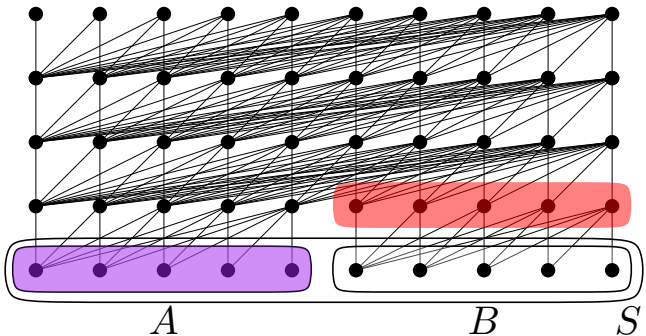
Flip-Breakability: Example



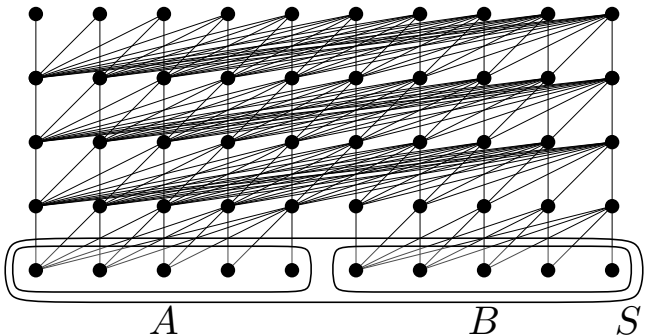
Flip-Breakability: Example



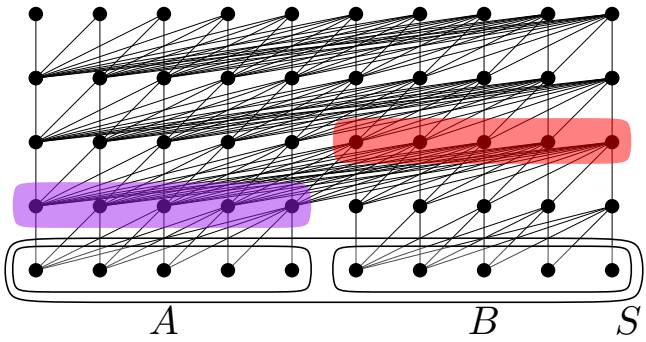
Flip-Breakability: Example



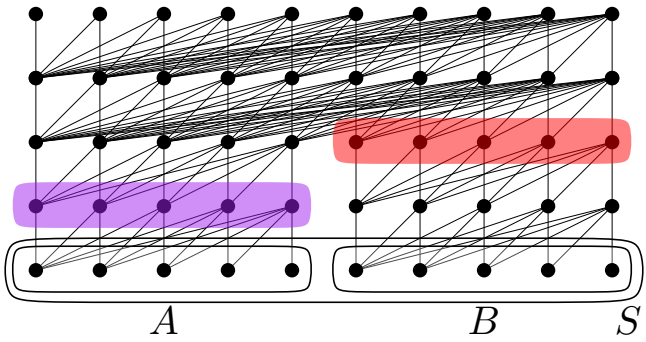
Flip-Breakability: Example



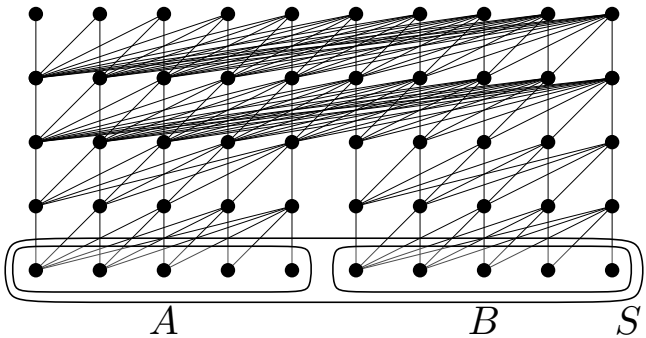
Flip-Breakability: Example



Flip-Breakability: Example



Flip-Breakability: Example



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 NIP**
	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 NIP**
	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 NIP**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-	**bd. shrubdepth**	**bd. cliquewidth**
	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 NIP**
	deletion-	**nowhere denseness**	**nowhere denseness**
dist-∞	flip-	**bd. shrubdepth**	**bd. cliquewidth**
	deletion-	**bd. treedepth**	**bd. treewidth**

Roadmap: FO Model Checking for Monadically Stable Classes

flip-flatness $\rightarrow$ flipper game $\rightarrow$ model checking

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: A contains a large independent set.

$\rightarrow$ B 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: A contains a large independent set.

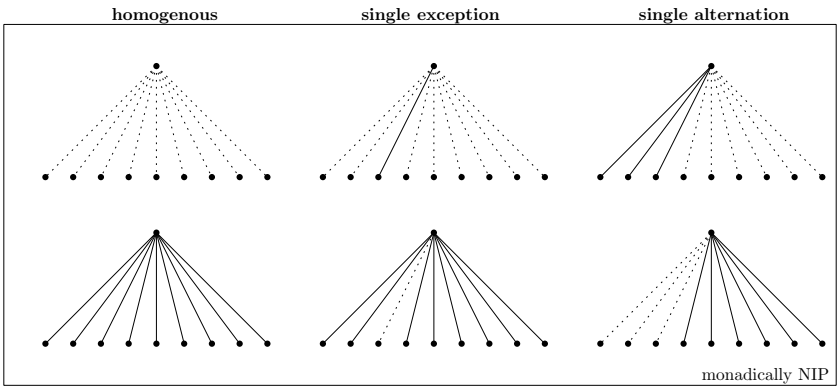
$\rightarrow$ B is distance-1 independent without performing any flips.

Case 2: A contains a large clique.

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

Indiscernibles

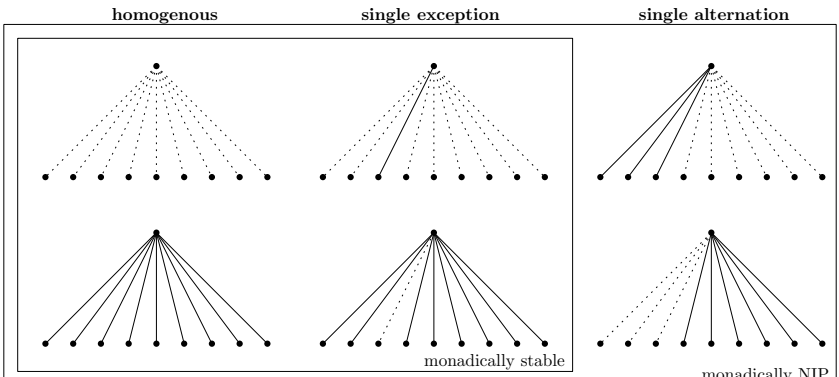
Every long sequence of vertices contains a still long subsequence that is *indiscernible*. In a monadically NIP class every vertex is connected to an indiscernible sequence in one of the following patterns:



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

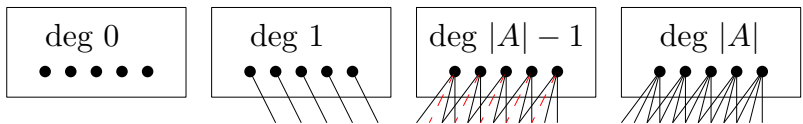
Indiscernibles

Every long sequence of vertices contains a still long subsequence that is *indiscernible*. In a monadically NIP class every vertex is connected to an indiscernible sequence in one of the following patterns:

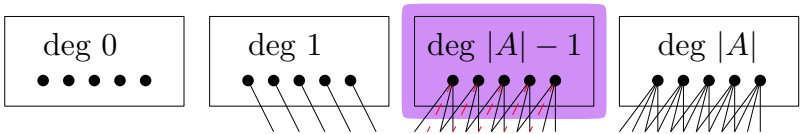


[Blumensath, 2011], [Dreier, Mählmann, Toruńczyk, Siebertz, 2023]

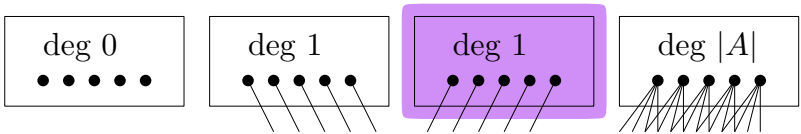
Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$

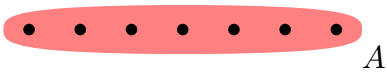


Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$

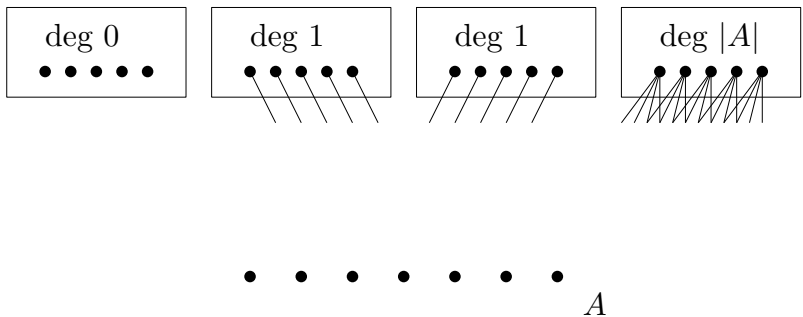


Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$

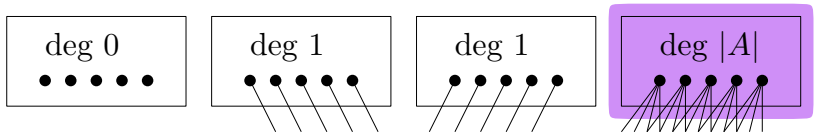




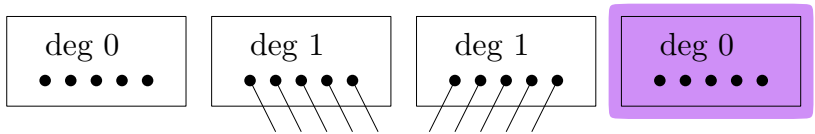
Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



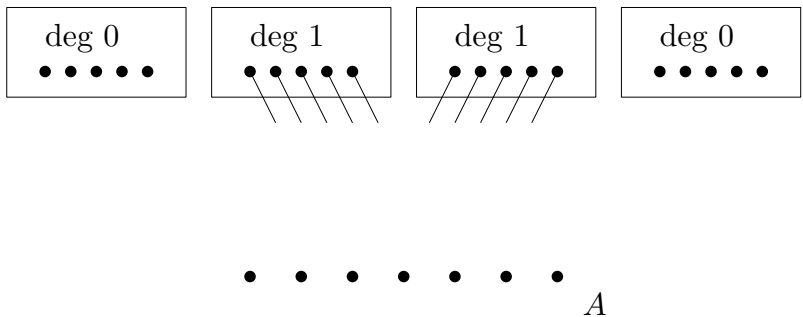
Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



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

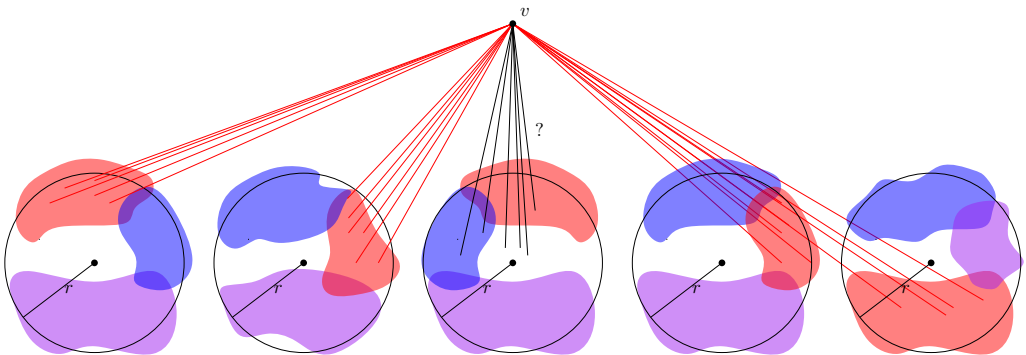


Monadic Stability $\Rightarrow$ Flip-Flatness: $r = 2$



Monadic Stability $\Rightarrow$ 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:



Roadmap: FO Model Checking for Monadically Stable Classes

flip-flatness $\checkmark \rightarrow$ flipper game $\rightarrow$ model checking

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. Connector chooses G_{i+1} as a radius-r ball in $G_i - v$.

Splitter wins once G_i has size 1.

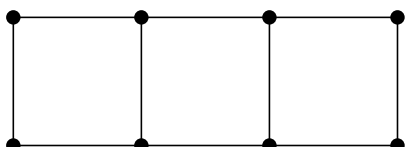
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



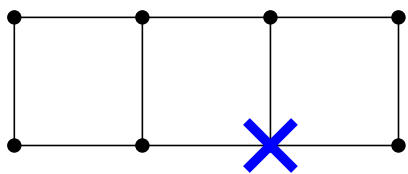
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



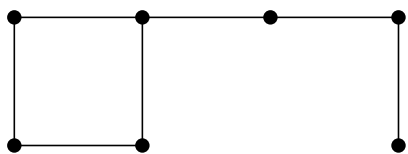
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



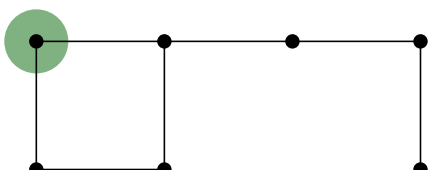
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



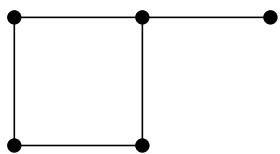
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



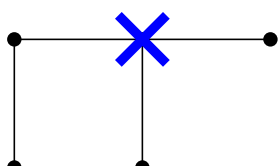
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



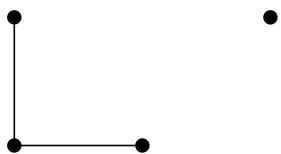
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



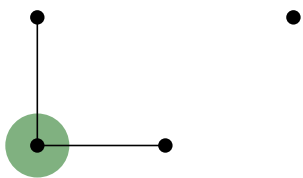
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



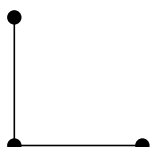
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



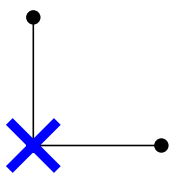
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



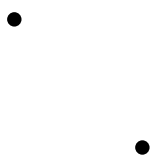
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



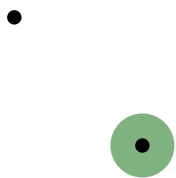
Splitter Game

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

1. Splitter chooses a vertex v to delete
2. Connector 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:



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. Connector 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 class of graphs $\mathcal{C}$ 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 class of graphs $\mathcal{C}$ 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.

Splitters strategy is efficiently computable and a main ingredient of the nowhere dense model checking.

The Splitter Game in Nowhere Dense Classes

Theorem [Grohe, Kreutzer, Siebertz, 2013]

A class of graphs $\mathcal{C}$ 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.

Splitters strategy is efficiently computable and a main ingredient of the nowhere dense model checking.

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

Flipper Game

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

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

Splitter wins once G_i has size 1.

Flipper Game

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

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

Flipper wins once G_i has size 1.

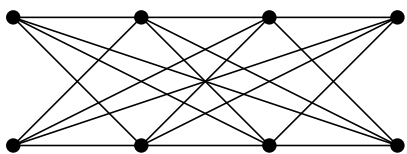
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



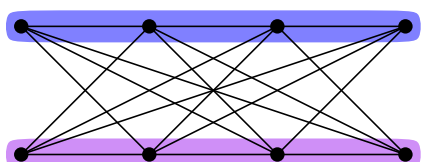
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



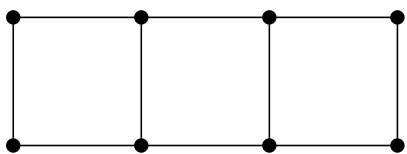
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



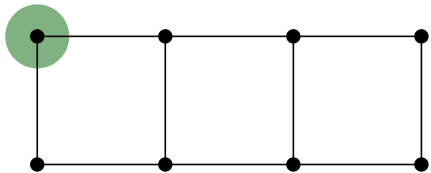
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



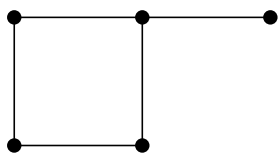
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



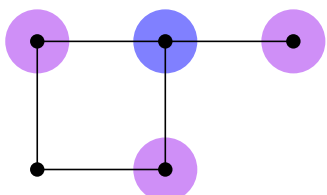
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



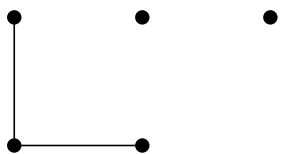
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



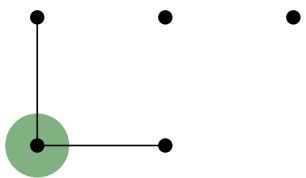
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



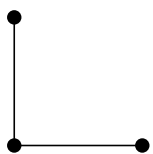
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



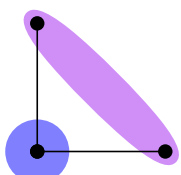
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



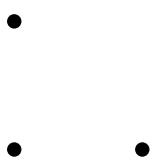
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



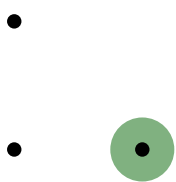
Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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:



Flipper Game

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

1. Flipper chooses a flip F
2. Connector 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

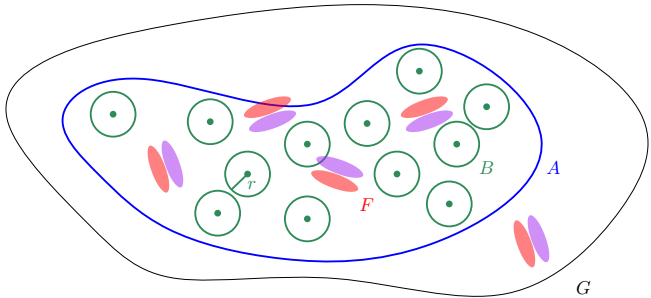
Theorem [Gajarský, Mählmann, McCarty, Ohlmann, Pilipczuk, Przybyszewski, Siebertz, Sokołowski, Toruńczyk, 2023]

A class of graphs $\mathcal{C}$ 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.

Flippers moves are computable in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.

Flip-Flatness



Definition (slightly informal) [Gajarský, Kreutzer]

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

Theorem [Dreier, Mählmann, Siebertz, Toruńczyk, 2023]

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

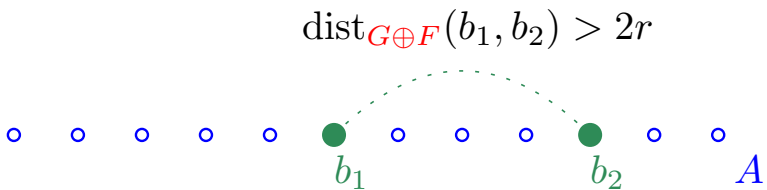
Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

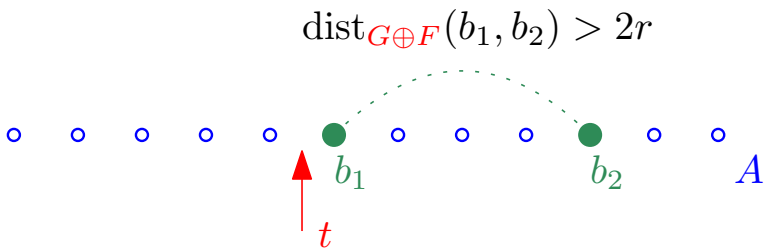
If the game continues long enough, we can apply flip-flatness to find a set $B \subseteq A$ which is $2r$-independent after applying constantly many flips F.



Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

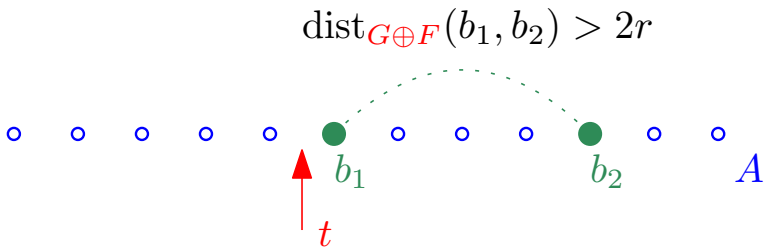
If the game continues long enough, we can apply flip-flatness to find a set $B \subseteq A$ which is $2r$-independent after applying constantly many flips F.



Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

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

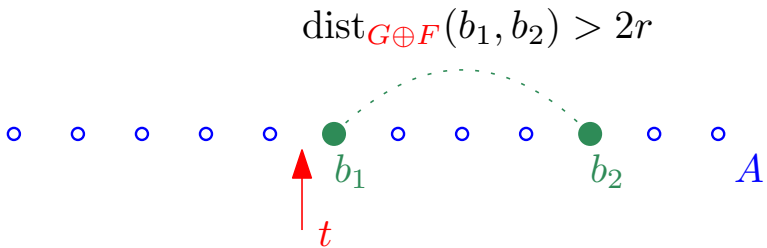


If Flipper had played the flip F at time t then only one of b_1 and b_2 could have survived in the graph.

Monadic Stability $\Rightarrow$ Flipper Wins - Proof Idea

Let $A = a_1, a_2, a_3, \ldots$ be the vertices played by Connector.

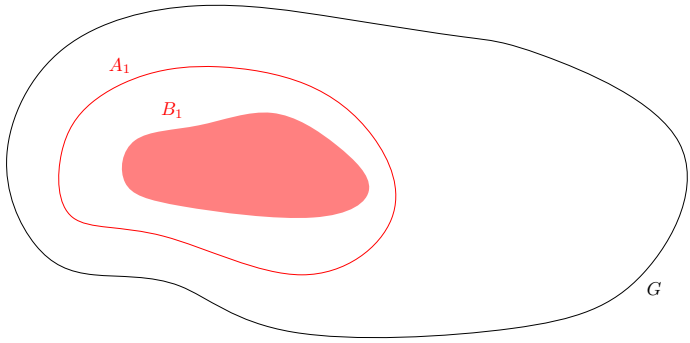
If the game continues long enough, we can apply flip-flatness to find a set $B \subseteq A$ which is $2r$-independent after applying constantly many flips F.



If Flipper had played the flip F at time t then only one of b_1 and b_2 could have survived in the graph.

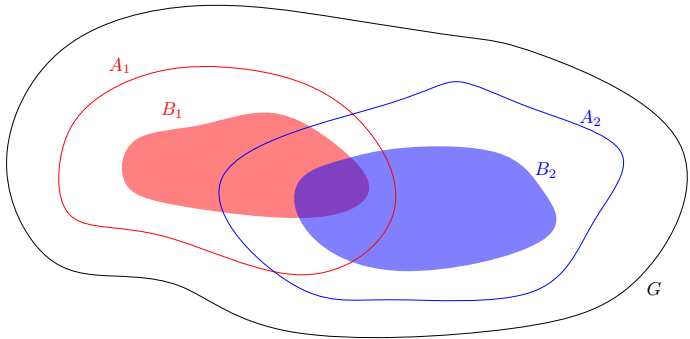
Problem: Flipper does not know A at time t.

Predictable Flip-Flatness



$\mathrm{ff}(A_1) = (B_1, F_1)$

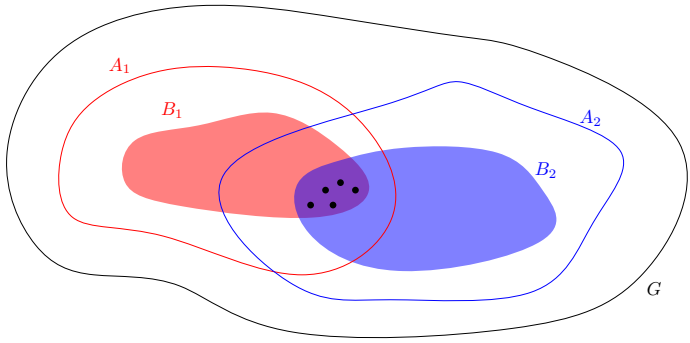
Predictable Flip-Flatness



$\mathrm{ff}(A_1) = (B_1, F_1)$

$\mathrm{ff}(A_2) = (B_2, F_2)$

Predictable Flip-Flatness



$\mathrm{ff}(A_1) = (B_1, F_1)$

$\mathrm{ff}(A_2) = (B_2, F_2)$

$|B_1 \cap B_2| \geq 5 \quad \Rightarrow \quad F_1 = F_2$

$F_1 = F_2$ are computable from a five-element subset of $B_1 \cap B_2$ in time $\mathcal{O}(n^2)$.

Flippers Winning Strategy

For every 5 element subset P of Connectors previous moves:

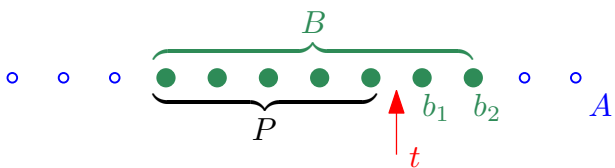
1. apply the flips $\mathrm{predict}(P)$ for radius $2r$
2. let Connector localize to an r-ball
3. undo $\mathrm{predict}(P)$

Flippers Winning Strategy

For every 5 element subset P of Connectors previous moves:

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

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

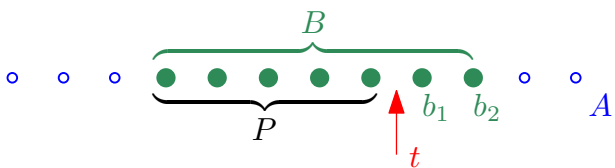


Flippers Winning Strategy

For every 5 element subset P of Connectors previous moves:

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

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



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

B was flipped $2r$-independent and only one of b_1, b_2 survived. Contradiction!

Roadmap: FO Model Checking for Monadically Stable Classes

flip-flatness $\checkmark \rightarrow$ flipper game $\checkmark \rightarrow$ model checking

Towards a Recursive Model Checking Algorithm

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

Towards a Recursive Model Checking Algorithm

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.

Towards a Recursive Model Checking Algorithm

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.

Towards a Recursive Model Checking Algorithm

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 $G \models \varphi \iff G^+ \oplus F \models \hat{\varphi}$.

Towards a Recursive Model Checking Algorithm

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?

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 $U \in \mathcal{U}$.

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

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

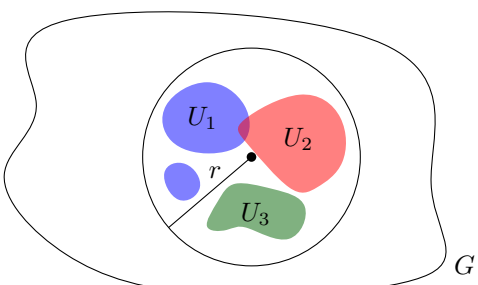
Guarded Formulas

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

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



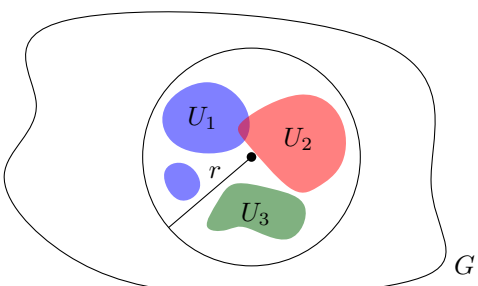
Guarded Formulas

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

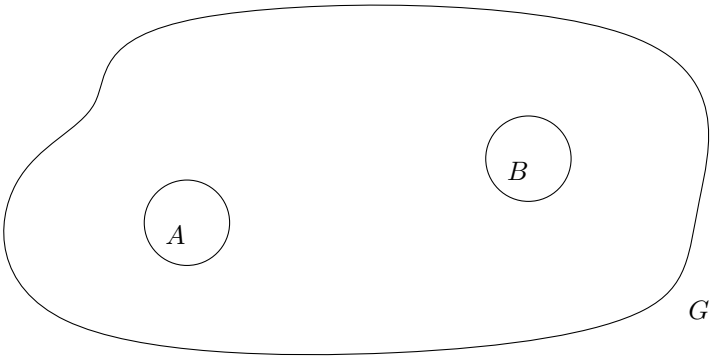
$$G \models \psi \iff G[U_1 \cup \ldots \cup U_t] \models \psi.$$



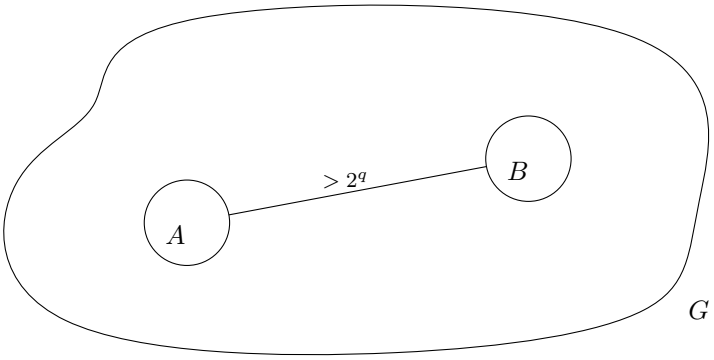
Goal: efficiently compute ψ s.t.

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.

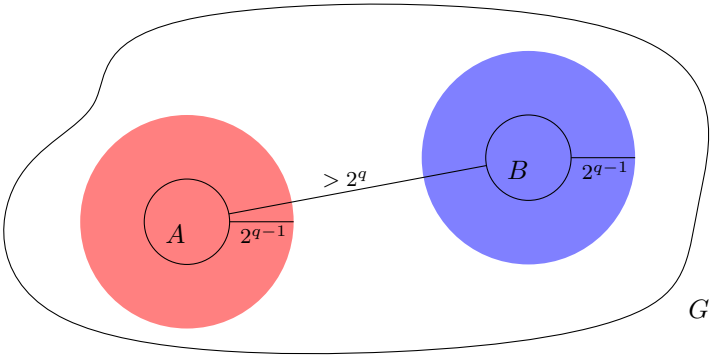
Local Types



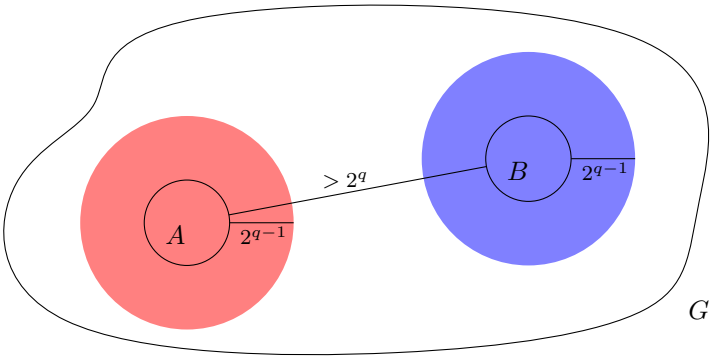
Local Types



Local Types

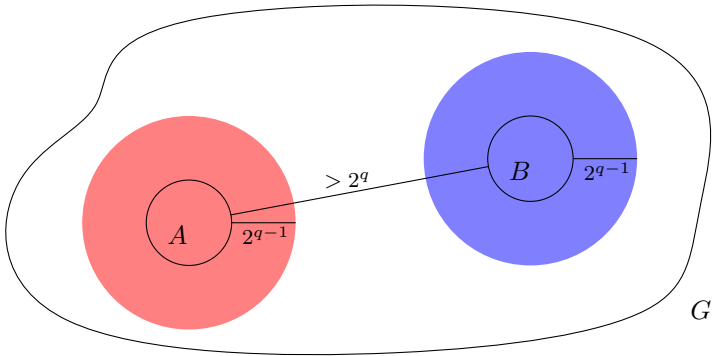


Local Types



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

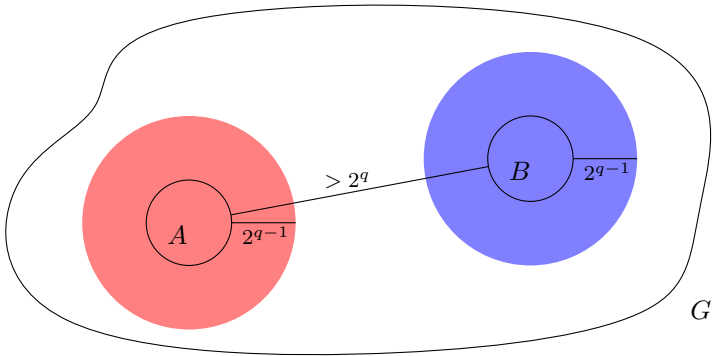
Local Types



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

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

Local Types

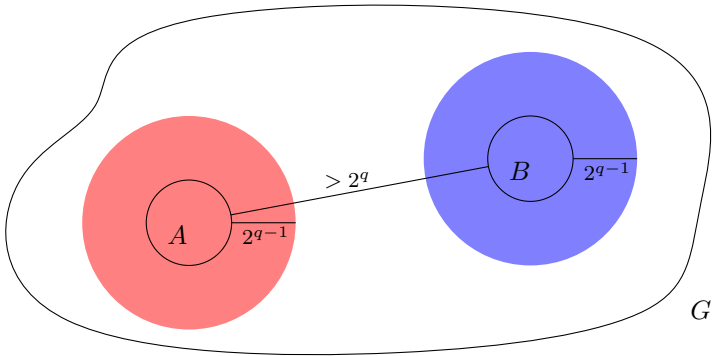


Assume $\mathrm{tp}_q(\bullet) = \mathrm{tp}_q(\bullet)$. $\quad \mathrm{tp}_q(G) := \{\psi : \psi$ has quantifier rank $\leq q$ and $G \models \psi\}$

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

There exists $u \in A$ with $G \models \psi(u)$ iff. there exists $v \in B$ with $G \models \psi(v)$.

Local Types



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

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

There exists $u \in A$ with $G \models \psi(u)$ iff. there exists $v \in B$ with $G \models \psi(v)$.

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

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

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

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 $|V(G)|$!

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

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 ✓

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 ✓

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 ✓

Recursion Tree

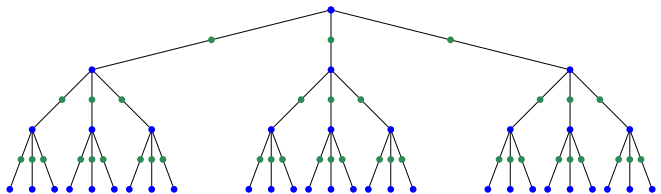
We can now play the Flipper game for radius 2^q:

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

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.

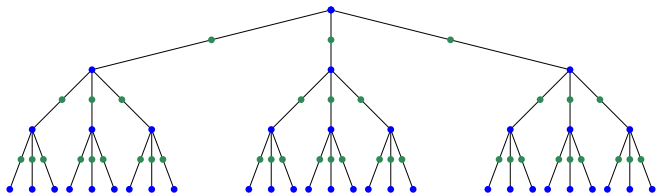


By monadic stability the depth of the recursion tree is bounded by $f(q)$.

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 $f(q)$.

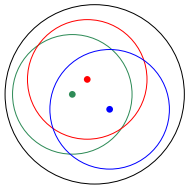
However the branching degree is n. This gives an $\mathcal{O}(n^{f(q)})$ algorithm.

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

Neighborhood Covers

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

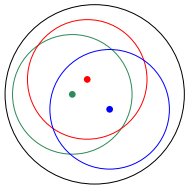
Idea: group neighborhoods that are close to each other into clusters.



Neighborhood Covers

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

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



Definition

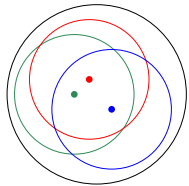
A family of sets $\mathcal{X}$ is a *neighborhood cover* with radius r, spread s, and degree d if

- 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.

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

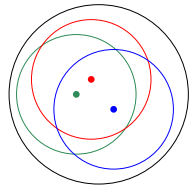
- each r-neighborhood of G is fully contained in one cluster $X \in \mathcal{X}$,
- each cluster is contained in an s-neighborhood of G,
- each vertex appears in at most d clusters.

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

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 in Monadically Stable Classes

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 in Monadically Stable Classes

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 $f(\varphi, \varepsilon) \cdot |V(G)|^{5+\varepsilon}$.

Roadmap: FO Model Checking for Monadically Stable Classes

flip-flatness ✓ $\rightarrow$ flipper game ✓ $\rightarrow$ model checking ✓

Map of the Universe