First-Order Model Checking on Structurally Sparse Graph Classes

Jan Dreier, Nikolas Mählmann, Sebastian Siebertz

STOC 2023

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.

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

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.

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

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.

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

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

Definition [Něsetřil, 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.

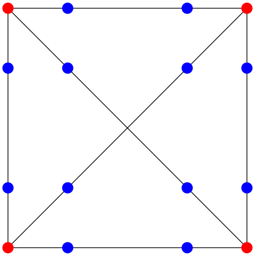


Figure: The 2-subdivided K_4.

Nowhere Dense Classes of Graphs

Definition [Něsetřil, 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, ...

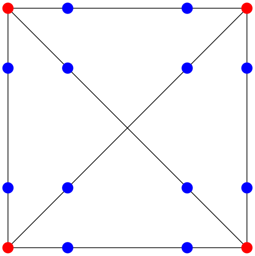


Figure: The 2-subdivided K_4.

Nowhere Dense Classes of Graphs

Definition [Něsetřil, 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, ...

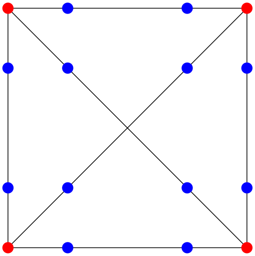


Figure: The 2-subdivided K_4.

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.

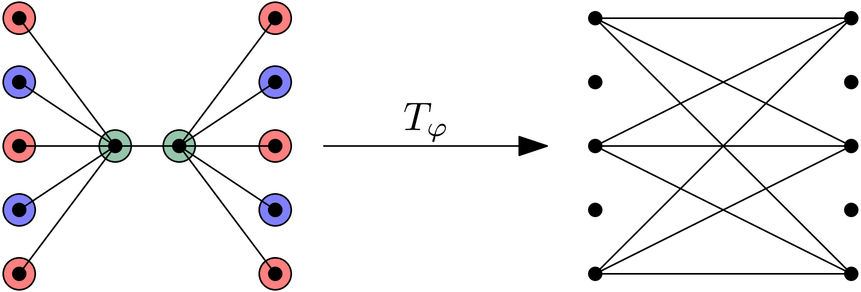
How to produce well behaved hereditary classes from sparse classes?

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 $\triangleq$ 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 [Gajarský, Kreutzer, Něsetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk, 2018], [Něsetřil, Ossona de Mendez, 2016]

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

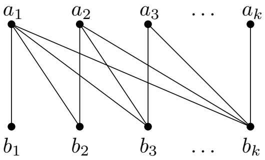
Structural Sparsity and Monadic Stability

Definition [Gajarský, Kreutzer, Něsetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk, 2018], [Něsetřil, Ossona de Mendez, 2016]

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 [Baldwin, Shelah, 1985]

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



$a_i \sim b_j \Leftrightarrow i \leq j$

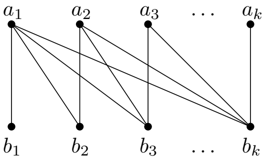
Structural Sparsity and Monadic Stability

Definition [Gajarský, Kreutzer, Něsetřil, Ossona de Mendez, Pilipczuk, Siebertz, Toruńczyk, 2018], [Něsetřil, Ossona de Mendez, 2016]

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 [Baldwin, Shelah, 1985]

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

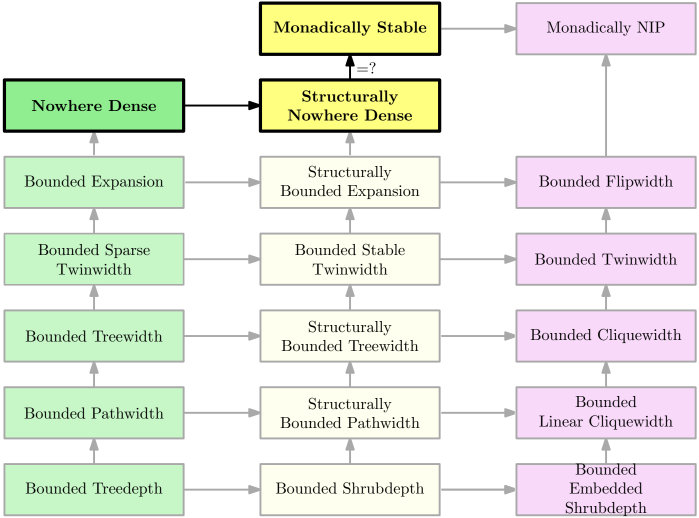


$a_i \sim b_j \Leftrightarrow i \leq j$

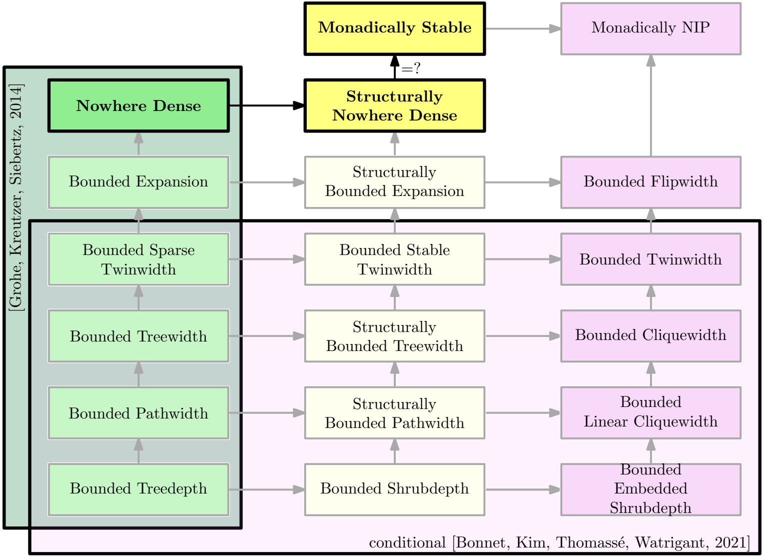
Every structurally nowhere dense class is monadically stable.

Conjecture: every monadically stable class is structurally nowhere dense.

Map of the Universe



Map of the Universe



Main Result

Theorem [Dreier, Mählmann, Siebertz]

Every structurally nowhere dense class admits FO model checking in time

$$f(\varphi) \cdot n^{11}.$$

Main Result

Theorem [Dreier, Mählmann, Siebertz]

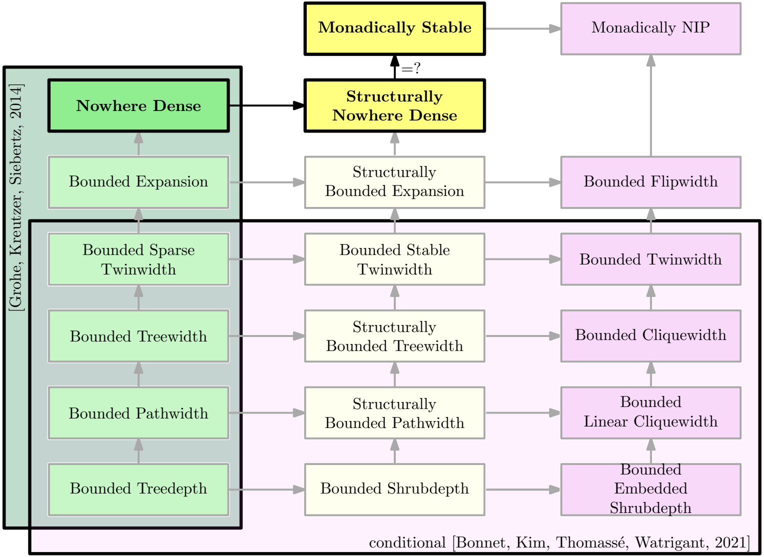
Every structurally nowhere dense class admits FO model checking in time

$$f(\varphi) \cdot n^{11}.$$

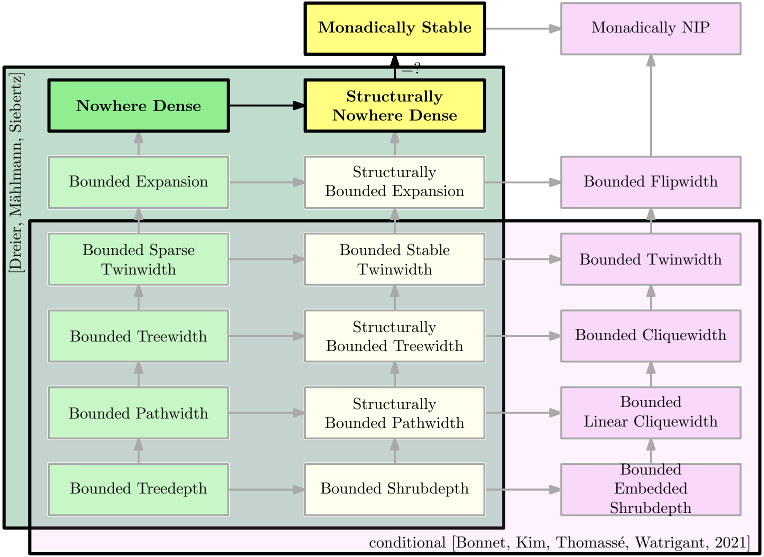
Theorem [Dreier, Mählmann, Siebertz]

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

Map of the Universe



Map of the Universe

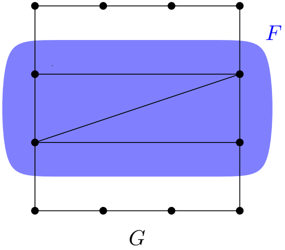


Flips

Denote by $G \oplus F$ the graph obtained from G by complementing edges between pairs of vertices from F.

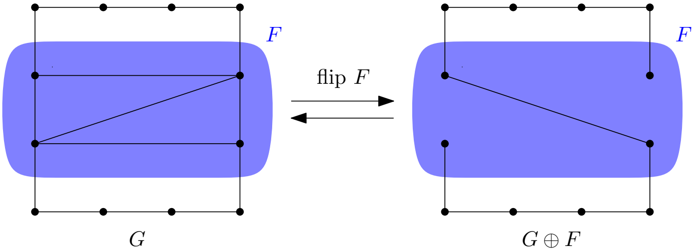
Flips

Denote by $G \oplus F$ the graph obtained from G by complementing edges between pairs of vertices from F.



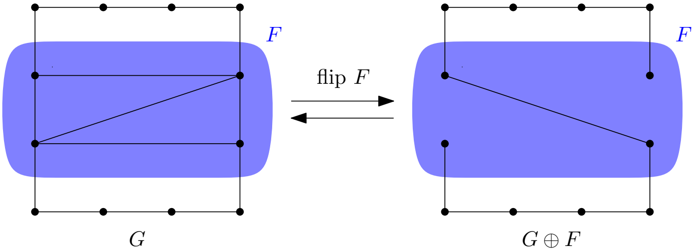
Flips

Denote by $G \oplus F$ the graph obtained from G by complementing edges between pairs of vertices from F.



Flips

Denote by $G \oplus F$ the graph obtained from G by complementing edges between pairs of vertices from F.



If we rewrite φ into $\hat{\varphi}$ such that

$$x \sim y \quad \text{is replaced with} \quad x \sim y \ \texttt{XOR} \ F(x) \wedge F(y)$$

then there exists a coloring G^+ of G such that

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

Flipper Game

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

1. Flipper chooses a flip set F

Flipper Game

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

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

Flipper Game

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

1. Flipper chooses a flip set 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.

Flipper Game

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

1. Flipper chooses a flip set 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:

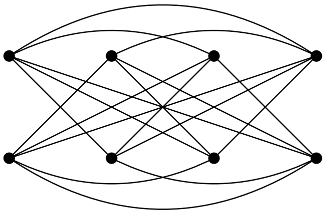
Flipper Game

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

1. Flipper chooses a flip set 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:



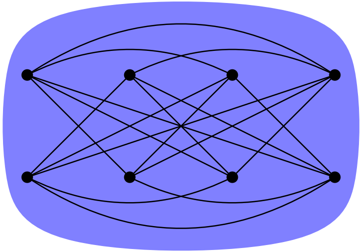
Flipper Game

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

1. Flipper chooses a flip set 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:



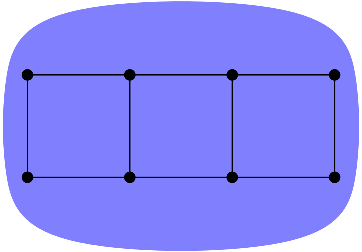
Flipper Game

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

1. Flipper chooses a flip set 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:



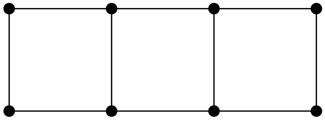
Flipper Game

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

1. Flipper chooses a flip set 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:



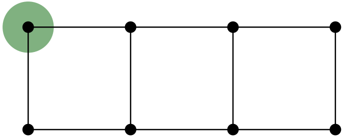
Flipper Game

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

1. Flipper chooses a flip set 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:



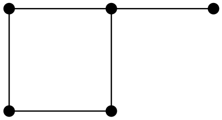
Flipper Game

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

1. Flipper chooses a flip set 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:



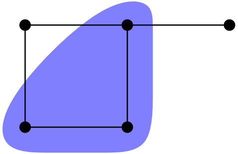
Flipper Game

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

1. Flipper chooses a flip set 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:



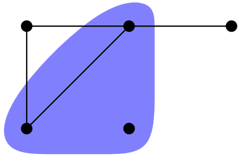
Flipper Game

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

1. Flipper chooses a flip set 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:



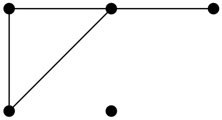
Flipper Game

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

1. Flipper chooses a flip set 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:



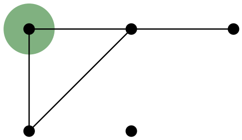
Flipper Game

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

1. Flipper chooses a flip set 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:



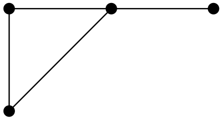
Flipper Game

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

1. Flipper chooses a flip set 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:



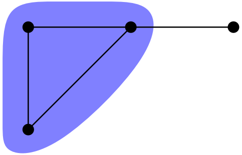
Flipper Game

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

1. Flipper chooses a flip set 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:



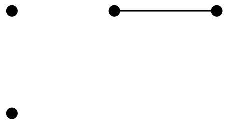
Flipper Game

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

1. Flipper chooses a flip set 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:



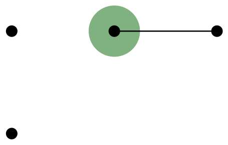
Flipper Game

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

1. Flipper chooses a flip set 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:



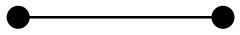
Flipper Game

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

1. Flipper chooses a flip set 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:



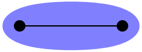
Flipper Game

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

1. Flipper chooses a flip set 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:



Flipper Game

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

1. Flipper chooses a flip set 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:

Flipper Game

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

1. Flipper chooses a flip set 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:

Flipper Game

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

1. Flipper chooses a flip set 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:

Flipper Game and Monadic Stability

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.

Flipper Game and Monadic Stability

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.

Moreover, Flippers moves can be computed in time $\mathcal{O}(n^2)$.

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

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

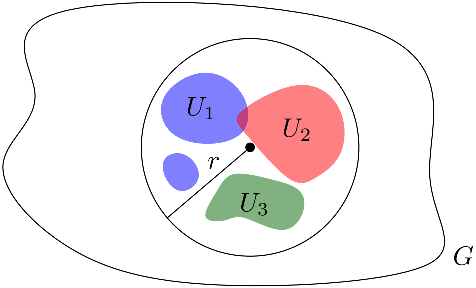
Guarded Formulas

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 $\{U_1, \dots, U_t\}$-guarded formula ψ we have

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



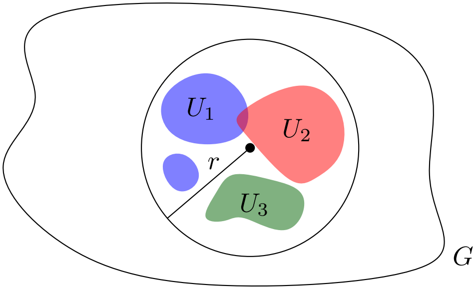
Guarded Formulas

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

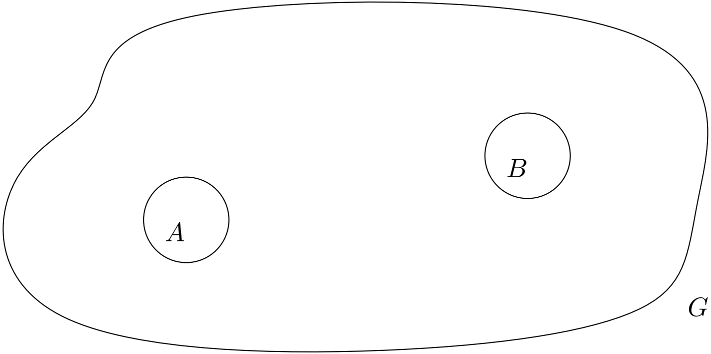
$$G \models \psi \iff G[U_1 \cup \ldots \cup U_t] \models \psi.$$



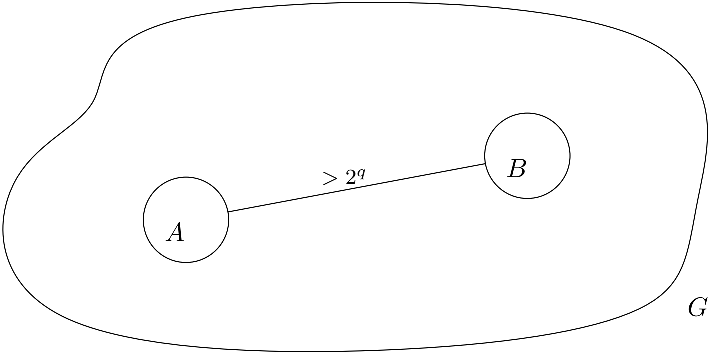
Goal: efficiently compute ψ s.t.

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.

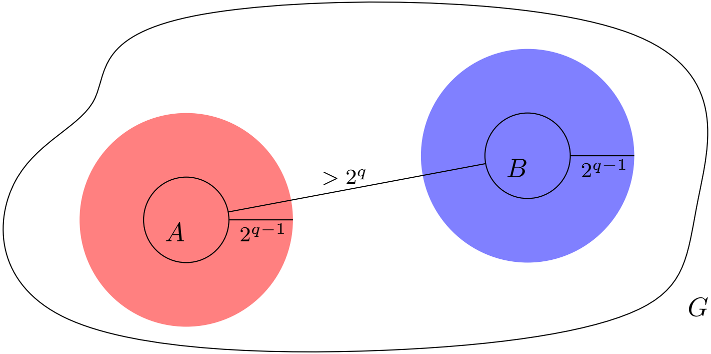
Local Types



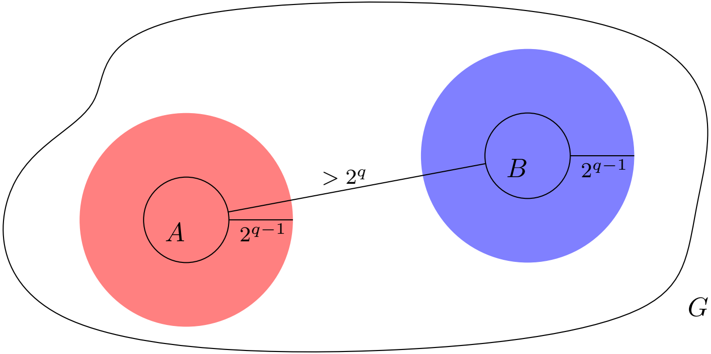
Local Types



Local Types

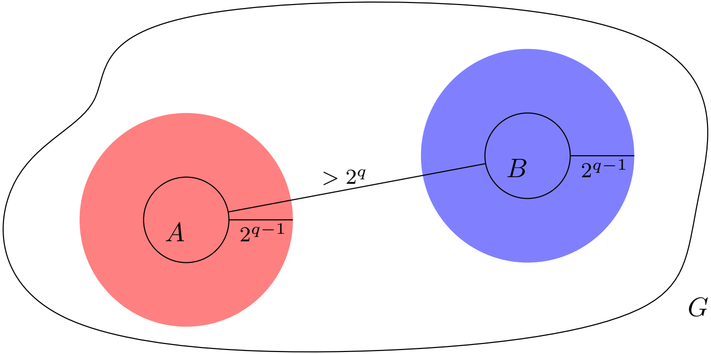


Local Types



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

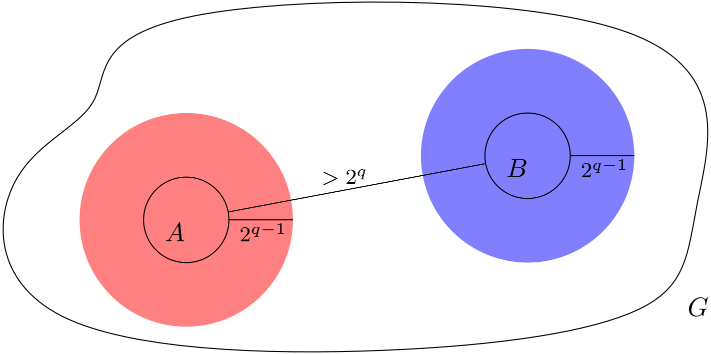
Local Types



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 $q - 1$.

Local Types

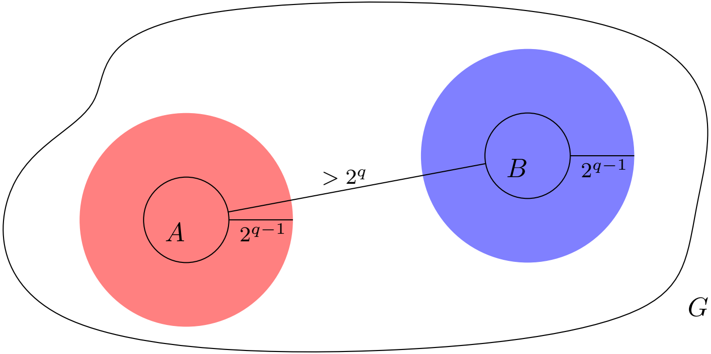


Assume $\mathrm{tp}_q(\textcolor{red}{\bullet}) = \mathrm{tp}_q(\textcolor{blue}{\bullet})$. $\mathrm{tp}_q(G) := \{\psi : \psi$ has quantifier rank $\leq q$ and $G \models \psi\}$

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

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

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

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

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

$|\mathcal{S}^\star|$ 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

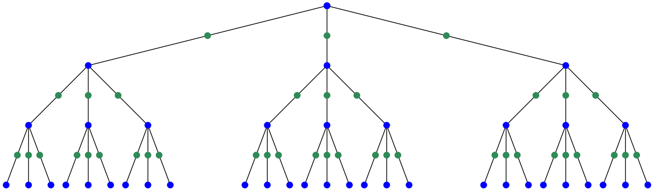
We can now play the Flipper game for radius 2^q:

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.

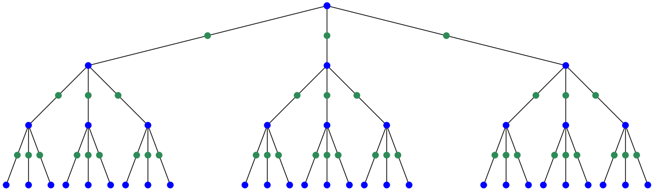


By monadic stability the depth of the recursion tree is bounded by $f(q)$.

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.

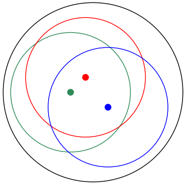


By monadic stability the depth of the recursion tree is bounded by $f(q)$.

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!

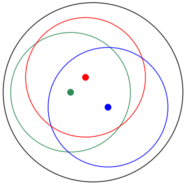
Neighborhood Covers



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!

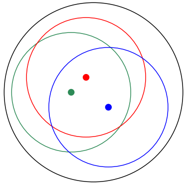
Idea: group neighborhoods that are close to each other into clusters.

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

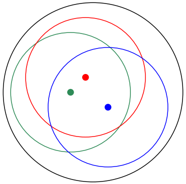
- each r-neighborhood of G is fully contained in one cluster $X \in \mathcal{X}$,
- each cluster is contained in an s-neighborhood of G,
- each vertex appears in at most d clusters.

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: $n^{1+\varepsilon'}$.

Approximating Sparse Neighborhood Covers

Theorem [Dreier, Mählmann, Siebertz]

Every structurally nowhere dense class admits sparse neighborhood covers.

Proof utilizes a treelike decomposition from [Dreier, Gajarský, Kiefer, Pilipczuk, Toruńczyk, 2022].

Approximating Sparse Neighborhood Covers

Theorem [Dreier, Mählmann, Siebertz]

Every structurally nowhere dense class admits sparse neighborhood covers.

Proof utilizes a treelike decomposition from [Dreier, Gajarský, Kiefer, Pilipczuk, Toruńczyk, 2022].

Theorem [Dreier, Mählmann, Siebertz]

Given a graph that admits a sparse neighborhood cover with radius r, spread s, and degree d. We can calculate a cover with radius r, spread s and degree $\mathcal{O}(\log(n)^2 \cdot d)$ in polynomial time.

Proof uses randomized rounding on an LP solution.

Main Result

Theorem [Dreier, Mählmann, Siebertz]

Every structurally nowhere dense class admits FO model checking in time

$$f(\varphi) \cdot |V(G)|^{11}.$$

Theorem [Dreier, Mählmann, Siebertz]

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

Main Result

Theorem [Dreier, Mählmann, Siebertz]

Every structurally nowhere dense class admits FO model checking in time

$$f(\varphi) \cdot |V(G)|^{11}.$$

Theorem [Dreier, Mählmann, Siebertz]

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

Conjecture

Every monadically stable class admits sparse neighborhood covers.