Algorithmic Meta-Theorems
Today last lecture, Friday was last exercise.

30 min overtime today.

Feedback on Tuwel is highly appreciated (short anonymous survey).

Oral exam: in January/February

There will be a question session before the oral exam.

Write email until end of year to dreier@ac.tuwien.ac.at if you want to participate in the oral exam.
Main Result for Sparse Graphs

Today, we finish the proof of the following theorem.

Theorem (Dvořák, Král, Thomas 2013)

Let $\mathcal{C}$ be a graph class with bounded expansion. There exists a function $f$ such that for every FO formula $\varphi$ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

And then briefly talk about dense graphs.
A functional graph $\vec{G}$ is a structure with signature 

$$\tau = \{h_1, h_2, \ldots, R_1, R_2, \ldots, Q_1, Q_2, \ldots\}$$

where

- $h_i : V \rightarrow V$ are unary functions
- $R_i \subseteq V$ are unary relations
- $Q_i \in \{0, 1\}$ are nullary relations
For a functional graph $\vec{G}$, the graph $\text{Gaifman}(\vec{G})$ has the same vertex set and edges $uv$ iff $h_i(u) = v$ or $h_i(v) = u$ for some $i$. 
For a functional graph $\vec{G}$, the graph $\text{Gaifman}(\vec{G})$ has the same vertex set and edges $uv$ iff $h_i(u) = v$ or $h_i(v) = u$ for some $i$.

We say a class of functional graphs has \textit{bounded expansion} if the class of their Gaifman graphs has.
\[ \vec{G} \exists y \, \varphi(y \bar{x}) \quad \rightarrow \quad \vec{G}' \varphi'(\bar{x}) \]
Quantifier Elimination

\[ \vec{G} \]

\[ \exists y \, \varphi(y \bar{x}) \]

\[ \rightarrow \]

\[ \vec{G}' \]

\[ \varphi'(\bar{x}) \]

Should still be short. Length depends only on \( \varphi \).
Quantifier Elimination

\[ \exists y \, \varphi(y\bar{x}) \]

Should still be sparse. Same Gaifman graph as \( \vec{G} \).

\[ \varphi'(\bar{x}) \]

Should still be short. Length depends only on \( \varphi \).
We say there is a **quantifier elimination procedure** for a class $C$ if the following holds.
We say there is a **quantifier elimination procedure** for a class $C$ if the following holds.

For every formula $\exists y \varphi(y\bar{x})$ where $\varphi$ is quantifier free there exists a quantifier-free formula $\varphi'(\bar{x})$ as follows:
We say there is a quantifier elimination procedure for a class $\mathcal{C}$ if the following holds.

For every formula $\exists y \varphi(y\vec{x})$ where $\varphi$ is quantifier free there exists a quantifier-free formula $\varphi'(\vec{x})$ as follows:

For every $\vec{G} \in \vec{\mathcal{C}}$ one can compute in time $O(|\vec{G}|)$ a functional graph $\vec{G}'$ with the same Gaifman graph as $\vec{G}$ and
We say there is a **quantifier elimination procedure** for a class $C$ if the following holds.

For every formula $\exists y \varphi(y, \bar{x})$ where $\varphi$ is quantifier free there exists a quantifier-free formula $\varphi'(\bar{x})$ as follows:

For every $\vec{G} \in \vec{C}$ one can compute in time $O(|\vec{G}|)$ a functional graph $\vec{G}'$ with the same Gaifman graph as $\vec{G}$ and

$$\vec{G} \models \exists y \varphi(y, \bar{v}) \iff \vec{G}' \models \varphi'(\bar{v}) \text{ for all } \bar{v} \in V(\vec{G})|\bar{x}|.$$
Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

$$\vec{G} \models \exists x_1 \left( \right)$$
If we have a quantifier elimination procedure, we can do model-checking.

\[ \tilde{G} \models \exists x_1 ( \forall x_2 ( \quad )) \]
If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G} \models \exists x_1 \left( \forall x_2 \left( \exists x_3 \right) \right) \]
If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G} \models \exists x_1 \left( \forall x_2 \left( \exists x_3 \left( \varphi(x_1 x_2 x_3) \right) \right) \right) \]
Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G} \models \exists x_1 \left( \forall x_2 \left( \exists x_3 \varphi(x_1 x_2 x_3) \right) \right) \]

replace with quantifier-free
Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G}' \models \exists x_1 \left( \forall x_2 \left[ \varphi'(x_1 x_2) \right] \right) \]
Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G}' \models \exists x_1 \left( \forall x_2 \varphi'(x_1 x_2) \right) \]

replace with quantifier-free

quantifier-free
If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G}''' \models \exists x_1 \{ \varphi''(x_1) \} \]
If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G}'' \models \exists x_1 \varphi''(x_1) \]

quantifier-free

replace with quantifier-free
Quantifier Elimination

If we have a quantifier elimination procedure, we can do model-checking.

\[ \vec{G}^{'''} \models \phi^{'''} \]
Roadmap: Construct quantifier elimination for graph classes of increasing complexity.

- forests of bounded depth ✔
- bounded treedepth
- bounded expansion
Let $\mathcal{C}$ be a class whose Gaifman graphs have *bounded treedepth*. Then $\mathcal{C}$ has a quantifier elimination procedure.
bounded treedepth

\[ \vec{G} \]

\[ \exists y \phi(y\overline{x}) \]
Bounded Treedepth

\[ \exists y \varphi(y \overline{x}) \quad \exists y \varphi'(\overline{x}) \]

bounded treedepth \[ \vec{G} \]

forest of bounded depth \[ \vec{G'} \]
Bounded Treedepth

bounded treedepth

$\vec{G}$

$\exists y \, \varphi(y\bar{x})$

forest of bounded depth

$\vec{G}''$

$\varphi''(\bar{x})$

Use previous theorem for quantifier elimination on forests.
Consider Gaifman graph of $\vec{G}$.

Graphs with treedepth $d$ contain no paths longer than $2d$.

We have a functional forest of depth at most $2d$.

All edges go between ancestors in the tree.

Encode edges of $\vec{G}$ by predicates at lower endpoints.

$P_{i,j}, \uparrow\downarrow(v) : \text{"There is edge from } v \text{ to parent } i(v) \text{ labeled with } j \text{ going upwards/downwards"}.$

This tree fully encodes $\vec{G}$.

Replace atoms $f_j(x) = y$ by guessing tree-relationship and checking the new predicates.
Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional.
Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. 

We have a functional forest of depth at most $2^d$. All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints. $P_{i,j,\uparrow}(v)$: "There is edge from $v$ to parent $i(v)$ labeled with $j$ going upwards". This tree fully encodes $\vec{G}$. Replace atoms $f_j(x) = y$ by guessing tree-relationship and checking the new predicates.
Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$. 

\hline

\begin{itemize}
  \item Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$. 
\end{itemize}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{tree_depth.png}
\caption{Diagram of a functional forest with treedepth $d$.}
\end{figure}
Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$.

All edges go between ancestors in the tree.
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$.

- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

$P_{i,j,\uparrow \downarrow}(v)$: “There is edge from $v$ to parent $i(v)$ labeled with $j$ going upwards/downwards”.
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$.

- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

  $P_{i,j,\uparrow}(v)$: “There is edge from $v$ to parent $i(v)$ labeled with $j$ going upwards/downwards”.

- This tree fully encodes $\vec{G}$. 
Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth $d$ contain no paths longer than $2^d$. We have a functional forest of depth at most $2^d$.

All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.

$P_{i,j,\uparrow}(v)$: “There is edge from $v$ to parent $i(v)$ labeled with $j$ going upwards/downwards”.

This tree fully encodes $\vec{G}$.

Replace atoms $f_j(x) = y$ by guessing tree-relationship and checking the new predicates.
Bounded Treedepth

\[ \exists y \varphi(y\overline{x}) \]
Bounded Treedepth

bounded treedepth

\[ \exists y \varphi(y\bar{x}) \]

forest of bounded depth

\[ \exists y \varphi'(\bar{x}) \]
Use previous theorem for quantifier elimination on forests.
Let $C$ be a class with bounded expansion. Then $C$ has a quantifier elimination procedure.
One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
Low Treedepth Colorings

One can label $G \in C$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in C$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in C$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
One can label $G \in \mathcal{C}$ with $f(p)$ colors such that every set of $p$ colors induces a graph with treedepth $\leq p$. 
We are given a formula $\exists y \varphi(\bar{x})$ where $\varphi$ is quantifier-free.
We are given a formula $\exists y \varphi(\bar{x})$ where $\varphi$ is quantifier-free.

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists a \ x_i \xrightarrow{g} a \land a \xrightarrow{\varphi} x_j$. 

Bounded Expansion
We are given a formula $\exists y \varphi(\bar{x})$ where $\varphi$ is quantifier-free.

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists a \ x_i \xrightarrow{g} a \land a \xrightarrow{\varphi} x_j$.

The result is a formula $\exists \bar{y} \psi(\bar{y}\bar{x})$ on a normal (non-functional) directed edge-labeled graph $G$. 
For every $\bar{v} \in V(G)^{|\bar{x}|}$

$G \models \exists \bar{y} \psi(\bar{y}\bar{v})$ iff
Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y} \psi(\bar{y} \bar{v}) \iff \bigvee S \subseteq \Lambda \ \text{and} \ \bar{G}^\prime \models \psi_S(\bar{y} \bar{v})$$

Let $\Lambda$ be the colors of a low-treedepth coloring of $G$ where every subgraph on $p = |\bar{y} \bar{x}|$ colors has treedepth $\leq p$. 
Bounded Expansion

For every $\bar{v} \in V(G)^{\bar{x}}$

$$G = \exists \bar{y} \psi(\bar{y} \bar{v}) \text{ iff }$$

$$\bigvee_{S \subseteq \Lambda \atop |S| = p} \text{colors}(\bar{v}) \subseteq S \wedge$$

Let $\Lambda$ be the colors of a low-treedepth coloring of $G$ where every subgraph on $p = |\bar{y}\bar{x}|$ colors has treedepth $\leq p$. 
Bounded Expansion

For every $\bar{v} \in V(G)^{\bar{x}}$

$$G = \exists \bar{y} \psi(\bar{y}\bar{v}) \text{ iff }$$

$$\bigvee_{S \subseteq \Lambda \atop |S| = p} \text{ colors}(\bar{v}) \subseteq S \land G[S] = \exists \bar{y} \psi(\bar{y}\bar{v}) \text{ iff }$$

Let $\Lambda$ be the colors of a low-treedepth coloring of $G$ where every subgraph on $p = |\bar{y}\bar{x}|$ colors has treedepth $\leq p$. 

17
Bounded Expansion

For every $\bar{v} \in V(G)^{|\bar{x}|}$

$$G \models \exists \bar{y} \psi(\bar{y} \bar{v}) \iff$$

$$\bigvee_{S \subseteq \Lambda \atop |S| = p} \text{colors}(\bar{v}) \subseteq S \land G[S] \models \exists \bar{y} \psi(\bar{y} \bar{v}) \iff$$

For all graphs $G[S]$ (of treedepth $\leq p$) we perform quantifier elimination.
For every \( \bar{v} \in V(G)^{|\bar{x}|} \)

\[
G \models \exists \bar{y} \psi(\bar{y}\bar{v}) \quad \text{iff} \quad \bigvee_{S \subseteq \Lambda, |S| = p} \text{colors}(\bar{v}) \subseteq S \wedge G[S] \models \exists \bar{y} \psi(\bar{y}\bar{v}) 
\]

\[
\bigvee_{S \subseteq \Lambda, |S| = p} \text{colors}(\bar{v}) \subseteq S \wedge \bar{G'}_S \models \psi_S(\bar{v}) \quad \text{iff} \quad \bar{G'}_S \models \bigvee_{S \subseteq \Lambda, |S| = p} \text{colors}(\bar{v}) \subseteq S \wedge \psi_S(\bar{v})
\]

For all graphs \( G[S] \) (of treedepth \( \leq p \)) we perform quantifier elimination.
Bounded Expansion

For every $\bar{v} \in V(G)^{\bar{x}}$

$$G \models \exists \bar{y} \psi(\bar{y}\bar{v}) \iff$$

$$\bigvee_{S \subseteq \Lambda} \operatorname{colors}(\bar{v}) \subseteq S \land G[S] \models \exists \bar{y} \psi(\bar{y}\bar{v}) \iff$$

$$\bigvee_{S \subseteq \Lambda} \operatorname{colors}(\bar{v}) \subseteq S \land \tilde{G'} \models \psi_S(\bar{v}) \iff$$

For all graphs $G[S]$ (of treedepth $\leq p$) we perform quantifier elimination. And stack the graphs on top of each other.
Bounded Expansion

For every \( \bar{v} \in V(G)^{|\bar{x}|} \)

\[
G \models \exists \bar{y} \psi(\bar{y}\bar{v}) \quad \text{iff} \quad \bigvee_{S \subseteq \Lambda \atop |S| = p} \text{colors}(\bar{v}) \subseteq S \land G[S] \models \exists \bar{y} \psi(\bar{y}\bar{v})
\]

iff

\[
\bigvee_{S \subseteq \Lambda \atop |S| = p} \text{colors}(\bar{v}) \subseteq S \land G'[S] \models \psi_S(\bar{v})
\]

iff

\[
G' \models \bigvee_{S \subseteq \Lambda \atop |S| = p} \text{colors}(\bar{v}) \subseteq S \land \psi_S(\bar{v}).
\]

Finally, pull out the graph.
This completes the proof. We have solved the model-checking problem on bounded expansion by performing quantifier elimination on trees and lifting it up using low treedepth colorings.
General rule: Things that work for bounded expansion also work for nowhere dense, but in an uglier way.
<table>
<thead>
<tr>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
</table>

- Enumerate $p$-subsets in time $(f(p))^p$.
- Enumerate $p$-subsets in time $(f(\varepsilon,p)\epsilon^p)^p = f(\varepsilon'/p,p)\epsilon'^p$.

- Do something very expensive in time $2^{2f(p)}$.
- Do something very expensive in time $2^{2f(\varepsilon/p,p)}\epsilon^p \geq 2^{n}$. 

- $2^{20}$
<table>
<thead>
<tr>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$-treedepth colorings with $f(p)$ colors</td>
<td>✓</td>
</tr>
</tbody>
</table>

Do something very expensive in time $2^{2^{f(p)}}$
### Comparison

<table>
<thead>
<tr>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$-treedepth colorings with $f(p)$ colors</td>
<td>$p$-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon &gt; 0$</td>
</tr>
<tr>
<td>Comparison</td>
<td>Bounded Expansion</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>( p )-treedepth colorings with ( f(p) ) colors ( \checkmark )</td>
</tr>
<tr>
<td>Enumerate ( p )-subsets in time ( \binom{f(p)}{p} ) ( \checkmark )</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$-treedepth colorings with $f(p)$ colors</td>
<td>$p$-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon &gt; 0$</td>
</tr>
<tr>
<td>Enumerate $p$-subsets in time $\binom{f(p)}{p}$</td>
<td>Enumerate $p$-subsets in time $\binom{f(\varepsilon, p)n^\varepsilon}{p}$</td>
</tr>
</tbody>
</table>

Do something very expensive in time $2^{2^{f(p)}}$

Do something very expensive in time $2^{2^{f(\varepsilon/p, p)n^\varepsilon}}$
<table>
<thead>
<tr>
<th>Comparison</th>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p$-treedepth colorings with $f(p)$ colors</td>
<td>$p$-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon &gt; 0$</td>
</tr>
<tr>
<td>Enumerate $p$-subsets</td>
<td>$(\binom{f(p)}{p})$</td>
<td>Enumerate $p$-subsets in time</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>$\binom{f(\varepsilon,p)n^\varepsilon}{p} \leq f(\varepsilon, p)n^{\varepsilon p}$</td>
</tr>
<tr>
<td>Bounded Expansion</td>
<td>Nowhere Dense</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>(p)-treedepth colorings with (f(p)) colors</td>
<td>(p)-treedepth colorings with (f(\varepsilon, p)n^{\varepsilon}) colors for all (\varepsilon &gt; 0) ✓</td>
<td></td>
</tr>
<tr>
<td>Enumerate (p)-subsets in time ((f(p))_p)</td>
<td>Enumerate (p)-subsets in time ((f(\varepsilon,p)n^\varepsilon)_p)</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>(\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}) ✓</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Bounded Expansion</th>
<th>Nowhere Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$-treedepth colorings with $f(p)$ colors</td>
<td>$p$-treedepth colorings with $f(\varepsilon, p)n^\varepsilon$ colors for all $\varepsilon &gt; 0$ ✓</td>
</tr>
<tr>
<td>Enumerate $p$-subsets in time $\binom{f(p)}{p}$</td>
<td>Enumerate $p$-subsets in time $\binom{f(\varepsilon, p)n^\varepsilon}{p}$</td>
</tr>
<tr>
<td>Do something very expensive in time $2^{2^{f(p)}}$</td>
<td>$\leq f(\varepsilon, p)n^{\varepsilon p} = f(\varepsilon'/p, p)n^{\varepsilon'}$ ✓</td>
</tr>
<tr>
<td>Bounded Expansion</td>
<td>Nowhere Dense</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>( p )-treedepth colorings with ( f(p) ) colors</td>
<td>( p )-treedepth colorings with ( f(\varepsilon, p)n^{\varepsilon} ) colors for all ( \varepsilon &gt; 0 )</td>
</tr>
<tr>
<td>Enumerate ( p )-subsets in time ( \binom{f(p)}{p} )</td>
<td>Enumerate ( p )-subsets in time ( \binom{f(\varepsilon, p)n^{\varepsilon}}{p} )</td>
</tr>
<tr>
<td>( \leq f(\varepsilon, p)n^{\varepsilon}p = f(\varepsilon'/p, p)n^{\varepsilon'} )</td>
<td></td>
</tr>
<tr>
<td>Do something very expensive in time ( 2^{2f(p)} )</td>
<td>Do something very expensive in time ( 2^{2f(\varepsilon/p, p)n^{\varepsilon}} )</td>
</tr>
<tr>
<td>Bounded Expansion</td>
<td>Nowhere Dense</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>( p )-treedepth colorings with ( f(p) ) colors</td>
<td>( p )-treedepth colorings with ( f(\varepsilon, p)n^\varepsilon ) colors for all ( \varepsilon &gt; 0 ) ✓</td>
</tr>
<tr>
<td>Enumerate ( p )-subsets in time ( \binom{f(p)}{p} ) ✓</td>
<td>Enumerate ( p )-subsets in time ( \binom{f(\varepsilon,p)n^\varepsilon}{p} ) ✓</td>
</tr>
<tr>
<td>Do something very expensive in time ( 2^{2^f(p)} ) ✓</td>
<td>Do something very expensive in time ( 2^{2^{f(\varepsilon/p,p)n^\varepsilon}} ) ✓</td>
</tr>
<tr>
<td></td>
<td>( \geq 2^{2^{\log(n)}} = 2^n ) ×</td>
</tr>
</tbody>
</table>
We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.
Beyond Nowhere Dense?

We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.

Are there any more interesting sparse graph classes beyond nowhere dense?
We have gradually defined more and more general sparse graph classes. The most general so far are nowhere dense classes.

Are there any more interesting sparse graph classes beyond nowhere dense?

We show the answer is no! We have found the most general sparse graph classes on which first-order problems are tractable!
A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.
A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.

**Theorem (Grohe, Kreuzer, Siebertz 2017)**

For every monotone graph class $\mathcal{C}$ holds $\mathcal{C}$ is nowhere dense iff the first-order model-checking problem on $\mathcal{C}$ is fpt (assuming $\text{FPT} \neq \text{AW}[\ast]$).
A graph class is *monotone* if it is closed under removing vertices or edges. This is a natural assumption for sparse graphs.

**Theorem (Grohe, Kreuzer, Siebertz 2017)**

For every monotone graph class $C$ holds $C$ is nowhere dense iff the first-order model-checking problem on $C$ is fpt (assuming $\text{FPT} \neq \text{AW[*]}$).

We discussed the forward implication already. We will prove the backward implication.
$H$ is an \textit{depth-}$r$ \textit{topological minor} of $G$ ($H \preceq^\text{top}_r G$) if $H$ can be built from $G$ by
$H$ is an \textit{depth-$r$ topological minor} of $G$ ($H \preceq^\text{top}_r G$) if $H$ can be built from $G$ by

- picking some \textit{nails},
$H$ is an *depth-$r$ topological minor* of $G$ ($H \preceq_{\text{top}}^r G$) if $H$ can be built from $G$ by

- picking some *nails*,
- picking internally vertex disjoint paths of length at most $2r + 1$ between nails,
$H$ is an depth-$r$ topological minor of $G$ ($H \preceq^\text{top}_r G$) if $H$ can be built from $G$ by

- picking some nails,
- picking internally vertex disjoint paths of length at most $2r + 1$ between nails,
- contracting the paths.
If $H$ is an depth-$r$ topological minor then it also is a depth-$r$ minor.
Topological Sparsity Measures

We can also measure sparsity using topological minors.
Topological Sparsity Measures

We can also measure sparsity using topological minors.

- \(\nabla_r(G) = \max\left\{ \frac{|E(H)|}{|V(H)|} \mid H \preceq_r G \right\}\)
- \(\omega_r(G) = \max\left\{ t \mid K_t \preceq_r G \right\}\)
We can also measure sparsity using topological minors.

- $\tilde{\nabla}_r(G) = \max\left\{ \frac{|E(H)|}{|V(H)|} \mid H \preceq^\text{top}_r G \right\}$
- $\tilde{\omega}_r(G) = \max\{ t \mid K_t \preceq^\text{top}_r G \}$
We can also measure sparsity using topological minors.

- $\tilde{\nabla}_r(G) = \max\left\{ \frac{|E(H)|}{|V(H)|} \mid H \preceq^{\text{top}}_r G \right\}$
- $\tilde{\omega}_r(G) = \max\left\{ t \mid K_t \preceq^{\text{top}}_r G \right\}$

As we saw on the last slide

- $\tilde{\nabla}_r(G) \leq \nabla_r(G)$
- $\tilde{\omega}_r(G) \leq \omega_r(G)$
Topological Sparsity Measures

We can also measure sparsity using topological minors.

\[ \tilde{\nabla}_r(G) = \max \left\{ \frac{|E(H)|}{|V(H)|} \mid H \preceq^{\text{top}}_r G \right\} \]

\[ \tilde{\omega}_r(G) = \max \left\{ t \mid K_t \preceq^{\text{top}}_r G \right\} \]

As we saw on the last slide

\[ \tilde{\nabla}_r(G) \leq \nabla_r(G) \]

\[ \tilde{\omega}_r(G) \leq \omega_r(G) \]

Surprisingly, also

\[ \nabla_r(G) \leq 2^{d^2+3d+3} \left( \lceil \tilde{\nabla}_r(G) \rceil \right)^{(d+2)^2} \]

\[ \omega_r(G) \leq 1 + \left( \tilde{\omega}_{3r+1}(G) \right)^{2d+2} \]
Therefore it does not matter if we consider normal or topological shallow minors.

**Bounded Expansion**

A graph class $\mathcal{C}$ has bounded expansion if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\nabla_r(G) \leq f(r)$.

**Nowhere Dense**

A graph class $\mathcal{C}$ is nowhere dense if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\omega_r(G) \leq f(r)$.
Therefore it does not matter if we consider normal or topological shallow minors.

**Bounded Expansion**

A graph class $\mathcal{C}$ has bounded expansion if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\tilde{\nabla}_r(G) \leq f(r)$.

**Nowhere Dense**

A graph class $\mathcal{C}$ is nowhere dense if there exists a function $f(r)$ such that for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$ we have $\tilde{\omega}_r(G) \leq f(r)$.
We can slightly restate this.

### Nowhere Dense

A graph class $\mathcal{C}$ is nowhere dense if for all $r \in \mathbb{N}$ there exists $t \in \mathbb{N}$ such that for all $G \in \mathcal{C}$ we have $K_t \not\preceq_r G$. 

### Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if it is not nowhere dense: for some $r \in \mathbb{N}$ there exists $t \in \mathbb{N}$ such that for all $G \in \mathcal{C}$ we have $K_t \not\preceq_r G$. 

We can slightly restate this.

**Nowhere Dense**

A graph class $C$ is nowhere dense if for all $r \in \mathbb{N}$ there exists $t \in \mathbb{N}$ such that for all $G \in C$ we have $K_t \not\preceq^\text{top}_r G$.

We say a graph class is *somewhere dense* if it is not nowhere dense:

**Somewhere Dense**

A graph class $C$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in C$ with $K_t \preceq^\text{top}_r G$. 
Subdivisions of Arbitrary Subgraphs

We show the following.

**Lemma**

If a graph class is monotone and somewhere dense then first-order model-checking on this class is \( AW[^*] \)-hard.
We need Ramsey’s Theorem.

**Lemma**

For every $t, r \in \mathbb{N}$ there is $k \in \mathbb{N}$ such that an edge-coloring of $K_k$ with $r$ colors contains a monochromatic $K_t$. 
Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preceq_r^\text{top} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
A graph class \( C \) is somewhere dense if there exists an \( r \in \mathbb{N} \) such that for all \( t \in \mathbb{N} \) there exists \( G \in C \) with \( K_t \preceq_r^{\text{top}} G \).

- Let \( C \) be monotone, somewhere dense.
- Pick \( r \) s.t. for every \( t \) there is a graph in \( C \) with \( K_t \) as depth-\( r \) topological minor.
Subdivisions of Arbitrary Subgraphs

**Somewhere Dense**

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preceq_{r}^{\text{top}} G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick $r$ s.t. for every $t$ there is a graph in $\mathcal{C}$ with $K_t$ as depth-$r$ topological minor.
- Using Ramsey, we can enforce that every path has length exactly $r' \leq 2r + 1$. 

$\in \mathcal{C}$
A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preceq_{\text{top}}^r G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick $r$ s.t. for every $t$ there is a graph in $\mathcal{C}$ with $K_t$ as depth-$r$ topological minor.
- Using Ramsey, we can enforce that every path has length exactly $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.
A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preceq_{\text{top}}^r G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick $r$ s.t. for every $t$ there is a graph in $\mathcal{C}$ with $K_t$ as depth-$r$ topological minor.
- Using Ramsey, we can enforce that every path has length exactly $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.
- $\mathcal{C}$ contains $r'$-subdivisions of arbitrary graphs.
Subdivisions of Arbitrary Subgraphs

Somewhere Dense

A graph class $\mathcal{C}$ is somewhere dense if there exists an $r \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ there exists $G \in \mathcal{C}$ with $K_t \preceq_r \text{top } G$.

- Let $\mathcal{C}$ be monotone, somewhere dense.
- Pick $r$ s.t. for every $t$ there is a graph in $\mathcal{C}$ with $K_t$ as depth-$r$ topological minor.
- Using Ramsey, we can enforce that every path has length exactly $r' \leq 2r + 1$.
- Since $\mathcal{C}$ is monotone, we can remove arbitrary edges and vertices.
- $\mathcal{C}$ contains $r'$-subdivisions of arbitrary graphs.
- We may even assume that the nails have some “hair” to distinguish them.
We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[*]-hard.
Reduction

We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[*]-hard.

- Assume we want to know whether $G \models \varphi$. 

Since the modified graph is in $C$, the modified formula can be evaluated in fpt time.
We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is $AW^{[*]}$-hard.

- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge $r'$ times and add hair. The result is in $C$.
We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[*]-hard.

- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge $r'$ times and add hair. The result is in $\mathcal{C}$.
- Replace
  - $x \sim y$ with $\text{dist}(x, y) \leq r'$.
  - $\exists x \xi$ with $\exists x \ deg(x) \geq 3 \land \xi$
  - $\forall x \xi$ with $\forall x \ deg(x) \geq 3 \rightarrow \xi$

Since the modified graph is in $\mathcal{C}$, the modified formula can be evaluated in fpt time.
We show that if one could do FPT first-order model-checking on a monotone, somewhere dense class, then one could do it on the class of all graphs. But there it is AW[∗]-hard.

- Assume we want to know whether $G \models \varphi$.
- Subdivide each edge $r'$ times and add hair. The result is in $C$.
- Replace
  - $x \sim y$ with $\text{dist}(x, y) \leq r'$.
  - $\exists x \xi$ with $\exists x \text{ degree}(x) \geq 3 \land \xi$
  - $\forall x \xi$ with $\forall x \text{ degree}(x) \geq 3 \rightarrow \xi$
- Since the modified graph is in $C$, the modified formula can be evaluated in fpt time.
For monotone graph classes, we cannot go beyond nowhere dense classes. Nowhere dense classes are a natural definition of “sparse graphs”.
Beyond Sparsity

This says nothing about dense tractable classes, such as the class of all cliques.
Beyond Sparsity

This says nothing about dense tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
This says nothing about dense tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
Beyond Sparsity

This says nothing about dense tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
Beyond Sparsity

This says nothing about *dense* tractable classes, such as the class of all cliques. A monotone tractable class containing all cliques contains all graphs.
Goal: Find tractable classes beyond sparsity.
Goal: Find tractable classes beyond sparsity.

Today, we discuss some results, as well as possible candidates.
Classes with FPT first-order model-checking?
Classes with FPT first-order model-checking?

- cliquewidth
- treewidth
- nowhere dense

- nowhere dense
- treewidth
- cliquewidth
Cliquewidth

Graphs of treewidth $w$ have cliquewidth at most $3 \cdot 2^w - 1$.

For a MSO$_1$ sentence $\phi$ and graph $G$, one can decide whether $G \models \phi$ in time $f(cw(G), |\phi|) n^3$ for some function $f$. 
Graphs of treewidth $\omega$ have cliquewidth at most $3 \cdot 2^{\omega-1}$.
Graphs of treewidth \( w \) have cliquewidth at most \( 3 \cdot 2^{w-1} \).

For a MSO\(_1\) sentence \( \varphi \) and graph \( G \) one can decide whether \( G \models \varphi \) in time \( f(\text{cw}(G), |\varphi|)n^3 \) for some function \( f \).
Classes with FPT first-order model-checking?
Classes with FPT first-order model-checking?

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

\[ \bar{G} \models \varphi \]
First-order model-checking is fpt on complements of nowhere dense classes by reduction.
First-order model-checking is fpt on complements of nowhere dense classes by reduction.
Classes with FPT first-order model-checking?
Can we do fpt model-checking on the class of fully bipartite graphs?
Can we do fpt model-checking on the class of fully bipartite graphs?

\[
G' \models \varphi'
\]

\[
G \models \varphi
\]
Can we do FPT model-checking on the class of fully bipartite graphs?

$G' \models \varphi'$

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

$G \models \varphi$
Can we do fpt model-checking on the class of fully bipartite graphs?

Obtain $\varphi'$ from $\varphi$ by replacing $x \sim y$ with $\text{dist}(x, y) = 3$

Also restrict quantifiers to black vertices

$G' \models \varphi'$

$G \models \varphi$
Classes with FPT first-order model-checking?
Classes with FPT first-order model-checking?

- Cliquewidth
- Treewidth
- Complements of nowhere dense
- Transductions of nowhere dense
- Nowhere dense
An interpretation $I = (\mu(x, y), \nu(x))$ maps $G$ to $I(G)$. 

$G$

$I(G)$
An interpretation $I = (\mu(x, y), \nu(x))$ maps $G$ to $I(G)$.

- vertices of $I(G)$: $\{v : G \models \nu(v)\}$. 

\[ \begin{align*} G & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \q
An interpretation \( I = (\mu(x, y), \nu(x)) \) maps \( G \) to \( I(G) \).

- vertices of \( I(G) \): \( \{ v : G \models \nu(v) \} \).
- edges of \( I(G) \): \( \{ uv : G \models \mu(u, v) \} \).
Assume we want to know whether $I(G') \models \varphi$.
Assume we want to know whether $I(G') \models \varphi$. We can build a formula $\varphi'$ from $I$ and $\varphi$ and instead evaluate $G \models \varphi'$. 

\[ G \models \varphi' \]

\[ I(G') \models \varphi \]
Assume we want to know whether $I(G') \models \varphi$. We can build a formula \( \varphi' \) from \( I \) and \( \varphi \) and instead evaluate $G \models \varphi'$.

Instead of asking whether $x \sim y$ in $I(G)$, ask whether $\mu(x, y)$ in $G$. 

\[
G \models \varphi' \quad \iff \quad I(G') \models \varphi
\]
A class $C$ is a *transduction* of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$. 

Example:
- $C$ is class of complete bipartite graphs
- $D$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x,y) = 3)$ such that if for every complete bipartite $G$ there is a tree $H$ and a coloring $H'$ of $H$ such that $G = I(H')$. 

A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation $I$ such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring $H'$ of $H$ such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees
A class $C$ is a *transduction* of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

Example:

- $C$ is class of complete bipartite graphs
- $D$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
A class $C$ is a *transduction* of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

Example:

- $C$ is class of complete bipartite graphs
- $D$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite $G$
A class $\mathcal{C}$ is a transduction of a class $\mathcal{D}$ if there exists an interpretation $I$ such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring $H'$ of $H$ such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite $G$
- there is a tree $H$
A class $\mathcal{C}$ is a *transduction* of a class $\mathcal{D}$ if there exists an interpretation $I$ such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring $H'$ of $H$ such that $G = I(H')$.

**Example:**

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite $G$
- there is a tree $H$
- and a coloring $H'$ of $H$
A class $\mathcal{C}$ is a transduction of a class $\mathcal{D}$ if there exists an interpretation $I$ such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring $H'$ of $H$ such that $G = I(H')$.

Example:

- $\mathcal{C}$ is class of complete bipartite graphs
- $\mathcal{D}$ is class of trees
- There is $I = (\text{black}(x), \text{dist}(x, y) = 3)$
- such that if for every complete bipartite $G$
- there is a tree $H$
- and a coloring $H'$ of $H$
- such that $G = I(H')$. 
Classes with FPT first-order model-checking?

Diagram:
- cliquewidth
- treewidth
- nowhere dense
- complements of nowhere dense
- transductions of nowhere dense
We believe that model-checking is fpt on transductions of nowhere dense classes.
Transductions

We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don’t know how to prove it.
We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don’t know how to prove it.

Assume we want to evaluate $I(G) \models \varphi$ with $G$ from a nowhere dense class.
We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don’t know how to prove it.

Assume we want to evaluate $I(G) \models \varphi$ with $G$ from a nowhere dense class. If we could construct $G$ from $I(G)$, we could instead evaluate $G \models \varphi'$ (where $\varphi'$ is obtained from $I$ and $\varphi$).
We believe that model-checking is fpt on transductions of nowhere dense classes.

But we don’t know how to prove it.

Assume we want to evaluate $I(G) |= \varphi$ with $G$ from a nowhere dense class. If we could construct $G$ from $I(G)$, we could instead evaluate $G |= \varphi'$ (where $\varphi'$ is obtained from $I$ and $\varphi$).

For some sparse graph classes, this approach works.
Let $\mathcal{C}$ be a transduction of a class with bounded degree. First-order model-checking is fpt on $\mathcal{C}$. 
Let $\mathcal{C}$ be a transduction of a class with bounded degree. First-order model-checking is fpt on $\mathcal{C}$.

Let $\mathcal{C}$ be a transduction of a class with locally bounded treewidth. First-order model-checking is fpt on $\mathcal{C}$.
Classes with FPT first-order model-checking?
Classes with FPT first-order model-checking?
You already know normal graphs.
You already know normal graphs.

In *trigraphs* there are additional red error edges.
We can contract two (not necessarily adjacent) vertices $a$ and $b$. The edges of the new vertex $ab$ follow this table.
We can contract two (not necessarily adjacent) vertices $a$ and $b$. The edges of the new vertex $ab$ follow this table.
A *contraction sequence* is a sequence of contractions until only a single vertex is left.
A contraction sequence is a sequence of contractions until only a single vertex is left.
A *contraction sequence* is a sequence of contractions until only a single vertex is left.

Example by Édouard Bonnet
A contraction sequence is a sequence of contractions until only a single vertex is left.
A contraction sequence is a sequence of contractions until only a single vertex is left.
A contraction sequence is a sequence of contractions until only a single vertex is left.
A *contraction sequence* is a sequence of contractions until only a single vertex is left.
A contraction sequence is a sequence of contractions until only a single vertex is left.
A contraction sequence is a sequence of contractions until only a single vertex is left.
Contraction Sequences

Bonnet, Kim, Thomassé, Watrigant 2021

**Twinwidth**: Smallest integer $d$ such there is a contraction sequence where the red degree is *at all times* at most $d$. 
Bonnet, Kim, Thomassé, Watrigant 2021

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

Bonnet, Kim, Thomassé, Watrigant 2021

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

Bonnet, Kim, Thomassé, Watrigant 2021
Bonnet, Kim, Thomassé, Watrigant 2021

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

---

Contraction Sequences
Contraction Sequences

Twinwidth: Smallest integer $d$ such there is a contraction sequence where the red degree is at all times at most $d$.‍
Bonnet, Kim, Thomassé, Watrigant 2021

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

Bonnet, Kim, Thomassé, Watrigant 2021

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

$abcdefghijkl$
The following classes have bounded twinwidth

- planar graphs,
- classes with bounded cliquewidth.
The following classes have bounded twinwidth

- planar graphs,
- classes with bounded cliquewidth.

The following classes do not have bounded twinwidth

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

**Bonnet, Kim, Thomassé, Watrigant 2021**

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

A class $C$ is a transduction of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

A graph class $C$ is monadically NIP if every transduction $D$ of $C$ is not the class of all graphs.

Conjecture: For every graph class $C$ that is closed under subgraphs holds:

First-order model-checking is fpt on $C$ iff $C$ is monadically NIP.
Classes with FPT first-order model-checking?

A class $C$ is a transduction of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

A graph class $C$ is monadically NIP if every transduction $D$ of $C$ is not the class of all graphs.

Conjecture: For every graph class $C$ that is closed under subgraphs holds: First-order model-checking is FPT on $C$ iff $C$ is monadically NIP.
A class $\mathcal{C}$ is a transduction of a class $\mathcal{D}$ if there exists an interpretation $I$ such that if for every $G \in \mathcal{C}$ there exists a graph $H \in \mathcal{D}$ and a coloring $H'$ of $H$ such that $G = I(H')$.

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

Conjecture

For every graph class $\mathcal{C}$ that is closed under subgraphs holds: First-order model-checking is fpt on $\mathcal{C}$ iff $\mathcal{C}$ is monadically NIP.
A class $C$ is a *transduction* of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

A graph class $C$ is *monadically NIP* if every transduction $D$ of $C$ is not the class of all graphs.
A class $C$ is a *transduction* of a class $D$ if there exists an interpretation $I$ such that if for every $G \in C$ there exists a graph $H \in D$ and a coloring $H'$ of $H$ such that $G = I(H')$.

A graph class $C$ is *monadically NIP* if every transduction $D$ of $C$ is not the class of all graphs.

**Conjecture**

For every graph class $C$ that is closed under subgraphs holds: First-order model-checking is fpt on $C$ iff $C$ is monadically NIP.