Algorithmic Meta-Theorems

192.122

WS21/22

Jan Dreier

dreier@ac.tuwien.ac.at

ALGORITHMS AND COMPLEXITY GROUP

TU WIEN Informatics

Minor Characterization

Minor-Free Graphs

A class of graphs $\mathcal{C}$ is *minor-free* if there exists a t such that every graph in $\mathcal{C}$ excludes K_t as a minor.

Minor Characterization

Minor-Free Graphs

A class of graphs $\mathcal{C}$ is *minor-free* if there exists a t such that every graph in $\mathcal{C}$ excludes K_t as a minor.

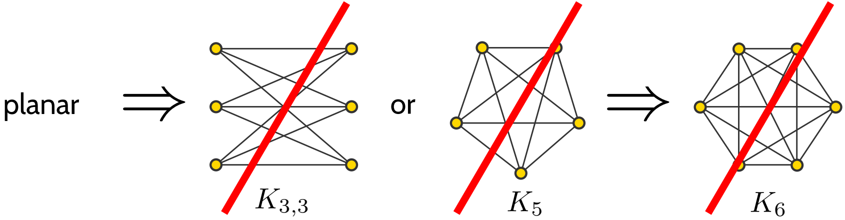
Every planar graph class and every class with bounded treewidth is minor-free.

Minor Characterization

Minor-Free Graphs

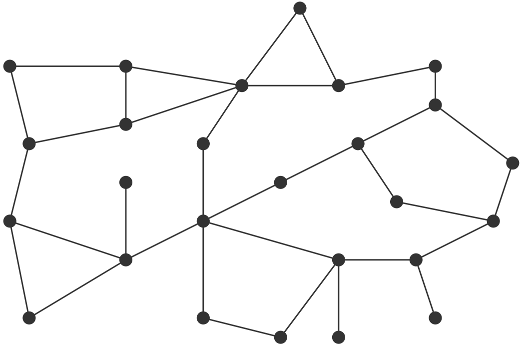
A class of graphs $\mathcal{C}$ is *minor-free* if there exists a t such that every graph in $\mathcal{C}$ excludes K_t as a minor.

Every planar graph class and every class with bounded treewidth is minor-free. Proof for planar graphs:



Shallow Minors

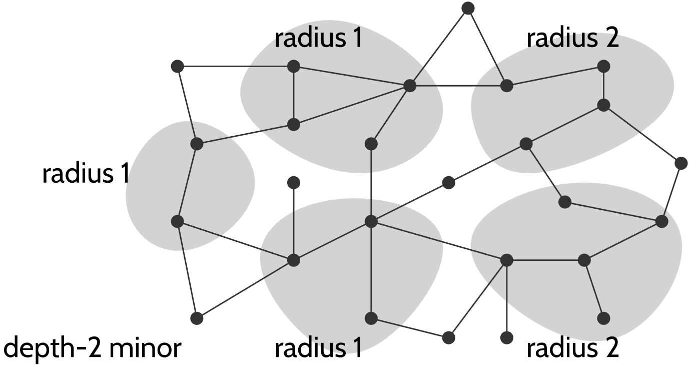
H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by



Shallow Minors

H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by

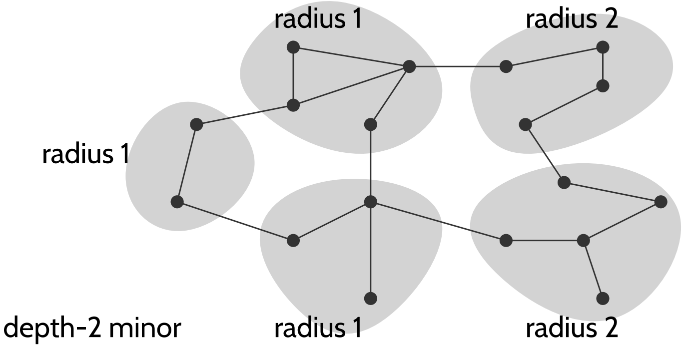
- picking some connected subgraphs with radius $\leq r$.



Shallow Minors

H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by

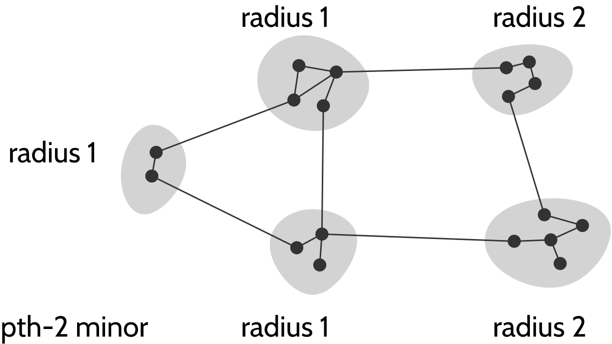
- picking some connected subgraphs with radius $\leq r$.
- removing all vertices outide these subgraphs



Shallow Minors

H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by

- picking some connected subgraphs with radius $\leq r$.
- removing all vertices outide these subgraphs
- merging each subgraph into a single vertex

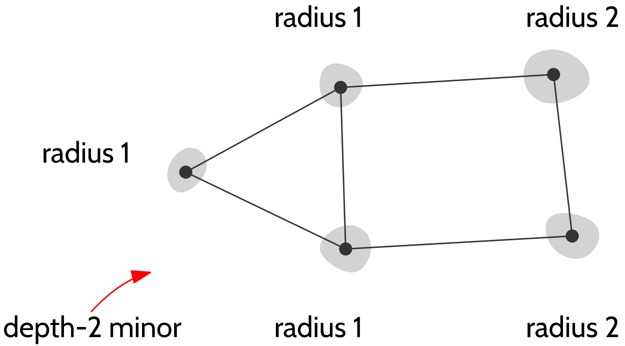


depth-2 minor

Shallow Minors

H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by

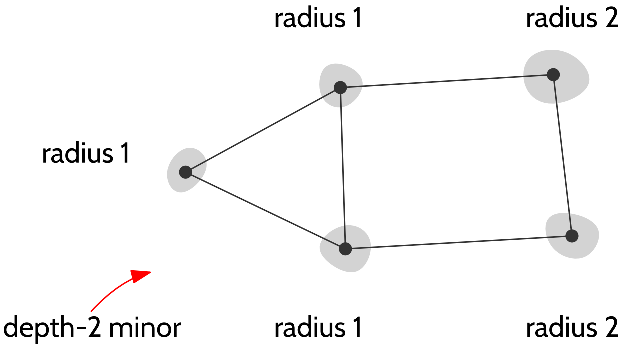
- picking some connected subgraphs with radius $\leq r$.
- removing all vertices outide these subgraphs
- merging each subgraph into a single vertex



Shallow Minors

H is an *depth-r minor* of G ($H \preccurlyeq_r G$) if H can be built from G by

- picking some connected subgraphs with radius $\leq r$.
- removing all vertices outide these subgraphs
- merging each subgraph into a single vertex
- removing edges



The Right Notion of Sparsity

We measure sparsity at depth r by measuring the depth-r minors of a graph G. This notion of sparsity was introduced by Nešetřil and Ossona de Mendez. We can think of two ways to do so

The Right Notion of Sparsity

We measure sparsity at depth r by measuring the depth-r minors of a graph G. This notion of sparsity was introduced by Nešetřil and Ossona de Mendez. We can think of two ways to do so

- bounding the average degree

$$\nabla_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r G\}$$

The Right Notion of Sparsity

We measure sparsity at depth r by measuring the depth-r minors of a graph G. This notion of sparsity was introduced by Nešetřil and Ossona de Mendez. We can think of two ways to do so

- bounding the average degree

$$\nabla_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r G\}$$

- bounding the clique size

$$\omega_r(G) = \max\{t \mid K_t \preccurlyeq_r G\}$$

The Right Notion of Sparsity

We measure sparsity at depth r by measuring the depth-r minors of a graph G. This notion of sparsity was introduced by Nešetřil and Ossona de Mendez. We can think of two ways to do so

- bounding the average degree

$$\nabla_r(G) = \max\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r G\}$$

- bounding the clique size

$$\omega_r(G) = \max\{t \mid K_t \preccurlyeq_r G\}$$

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 $\nabla_r(G) \leq f(r)$.

The Right Notion of Sparsity

We measure sparsity at depth r by measuring the depth-r minors of a graph G. This notion of sparsity was introduced by Nešetřil and Ossona de Mendez. We can think of two ways to do so

- bounding the average degree

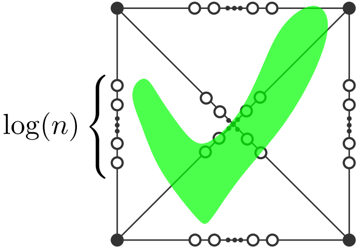
$$\nabla_r(G) = \max\left\{\frac{|E(H)|}{|V(H)|} \mid H \preccurlyeq_r G\right\}$$

- bounding the clique size

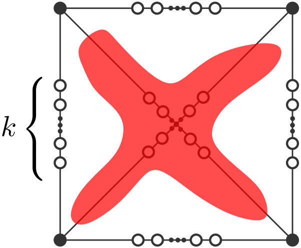
$$\omega_r(G) = \max\{t \mid K_t \preccurlyeq_r G\}$$

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 $\omega_r(G) \leq f(r)$.



clique of size n



clique of size n

Many Sparse Graph Classes

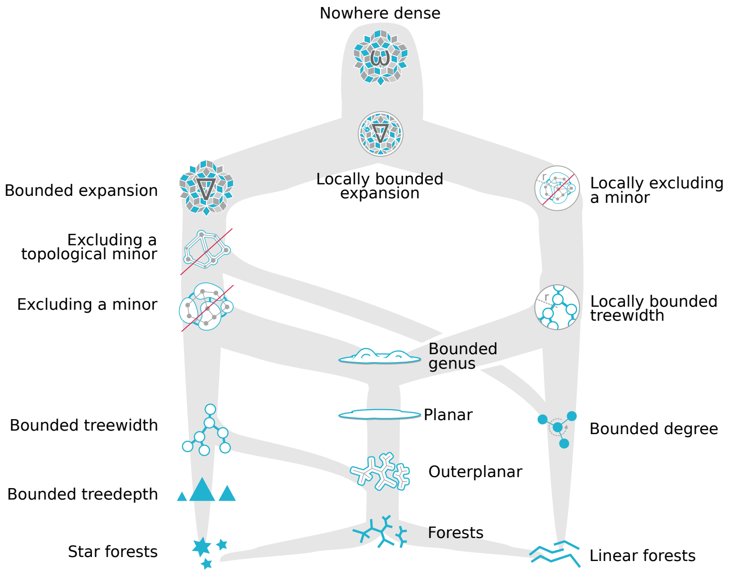


Figure by Felix Reidl

Main Results for Sparse Graphs

Theorem (Grohe, Kreuzer, Siebertz 2017)

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

Main Results for Sparse Graphs

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 φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

Main Results for Sparse Graphs

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 φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

Theorem (Grohe, Kreuzer, Siebertz 2017)

Let $\mathcal{C}$ be a nowhere dense graph class. There exists a function f such that for every $\varepsilon > 0$, FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(\varepsilon, |\varphi|)n^{1+\varepsilon}$.

Main Results for Sparse Graphs

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 φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

Theorem (Grohe, Kreuzer, Siebertz 2017)

Let $\mathcal{C}$ be a nowhere dense graph class. There exists a function f such that for every $\varepsilon > 0$, FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(\varepsilon, |\varphi|)n^{1+\varepsilon}$.

General rule: Things that work for bounded expansion also work for nowhere dense, but in an uglier way. This is why we focus on bounded expansion only in this course.

Existential Model-Checking

We will first prove a weaker result that is a building block in many other algorithms.

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

Existential Model-Checking

We will first prove a weaker result that is a building block in many other algorithms.

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

This is (more or less) equivalent to deciding in time $f(|H|)n$ whether a pattern graph H occurs as induced subgraph.

Existential Model-Checking

Proof of equivalence:

- Assume we want to know whether $G \models \varphi$ for some existential formula with q quantifiers. For example $\varphi = \exists x \exists y \exists z x{\sim}y \wedge y{\not\sim}z$.

Proof of equivalence:

- Assume we want to know whether $G \models \varphi$ for some existential formula with q quantifiers. For example $\varphi = \exists x \exists y \exists z x{\sim}y \wedge y{\not\sim}z$.
- Compute set $\mathcal{H}$ of all graphs with at most q vertices and $H \models \varphi$. In our case,

$\mathcal{H} = \{$  $\}$

Existential Model-Checking

Proof of equivalence:

- Assume we want to know whether $G \models \varphi$ for some existential formula with q quantifiers. For example $\varphi = \exists x \exists y \exists z x{\sim}y \wedge y{\not\sim}z$.
- Compute set $\mathcal{H}$ of all graphs with at most q vertices and $H \models \varphi$. In our case,

$\mathcal{H} = \{$  $\}$

- Now $G \models \varphi$ iff G contains some graph from $\mathcal{H}$ as induced subgraph.
  - Assume $G \models \varphi$. Then the satisfying assignment describes induced subgraph H of G with $H \models \varphi$.
  - Assume $H \in \mathcal{H}$ is induced subgraph of G. Then $H \models \varphi$. This does not change while adding the remaining vertices of G.

Alternative Characterizations

How can we prove these results?

- Gaifman does not help much because neighborhoods can be the whole graph.
- So far, all we know that certain shallow minors are not present.
- If we have a better understanding of the structure of sparse graphs, this will help us.

Alternative Characterizations

There are many alternative definitions of bounded expansion and nowhere dense classes.

- shallow minors
- generalized coloring numbers
- low treedepth colorings
- transitive fraternal augmentations
- quasi-wideness
- connector-splitter games

Which one is best depends on the task.

Low Treedepth Colorings

To prove the result, we will use the powerful notion of *low treedepth colorings*.

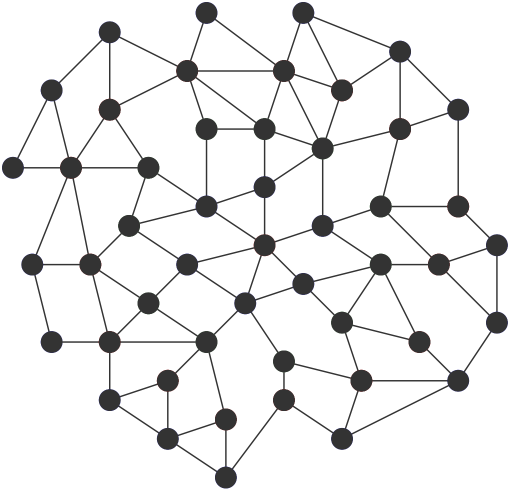
Low Treedepth Colorings

To prove the result, we will use the powerful notion of *low treedepth colorings*.

As a warmup, we solve the problem on planar graphs and then generalize the approach to bounded expansion.

Baker's Technique

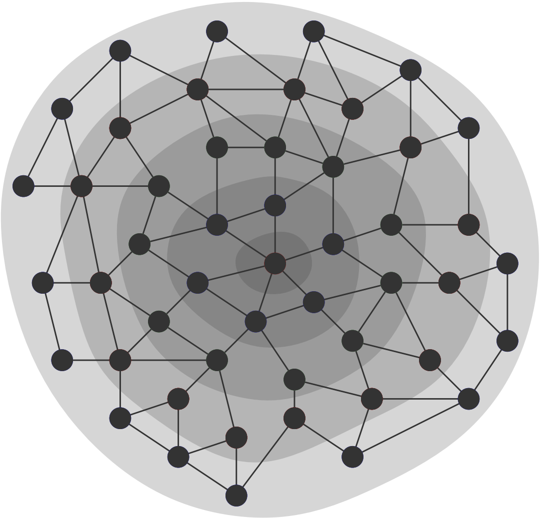
On any planar graph you can do the following.



Baker's Technique

On any planar graph you can do the following.

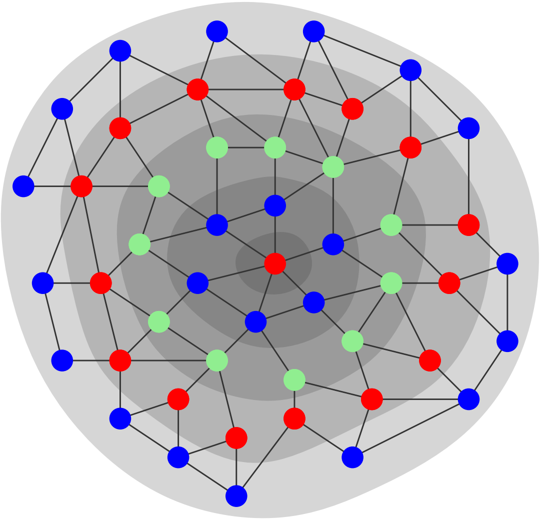
- do breadth-first search



Baker's Technique

On any planar graph you can do the following.

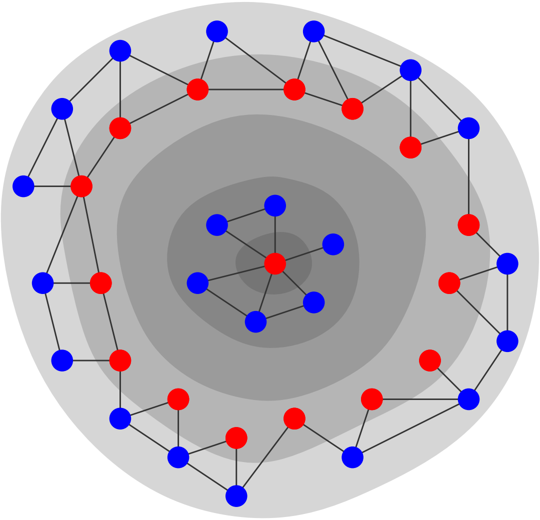
- do breadth-first search
- give layer i color $i \bmod p$



Baker's Technique

On any planar graph you can do the following.

- do breadth-first search
- give layer i color i mod p
- pick a strict subset of colors

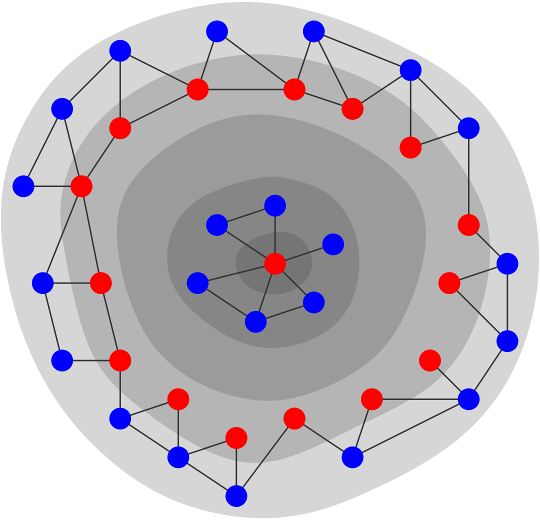


Baker's Technique

On any planar graph you can do the following.

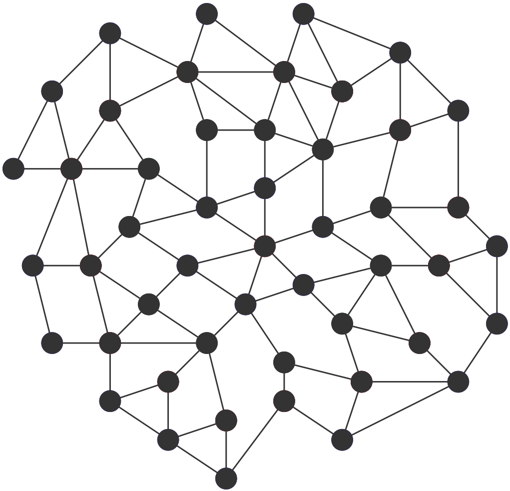
- do breadth-first search
- give layer i color i mod p
- pick a strict subset of colors

The resulting graph has treewidth at most $3p + 1$.



Baker's Technique

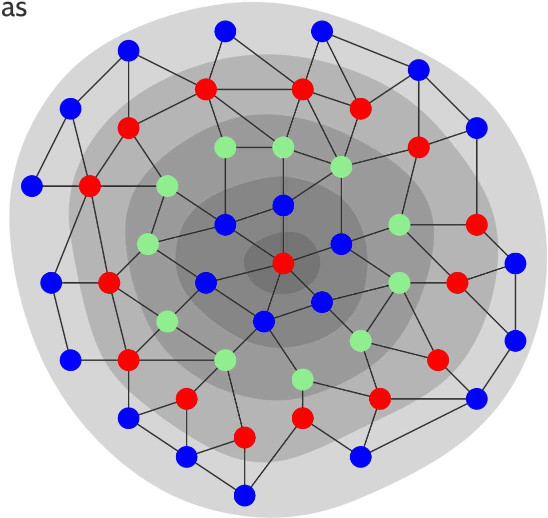
We want to know whether G has H as an induced subgraph.



Baker's Technique

We want to know whether G has H as an induced subgraph.

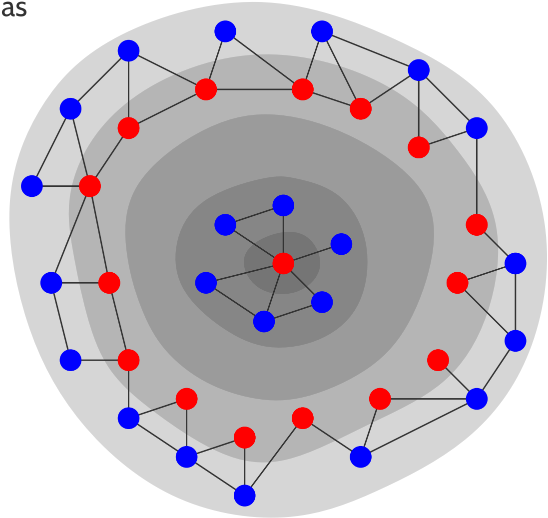
- color the graph as before with $p = |H| + 1$ colors.



Baker's Technique

We want to know whether G has H as an induced subgraph.

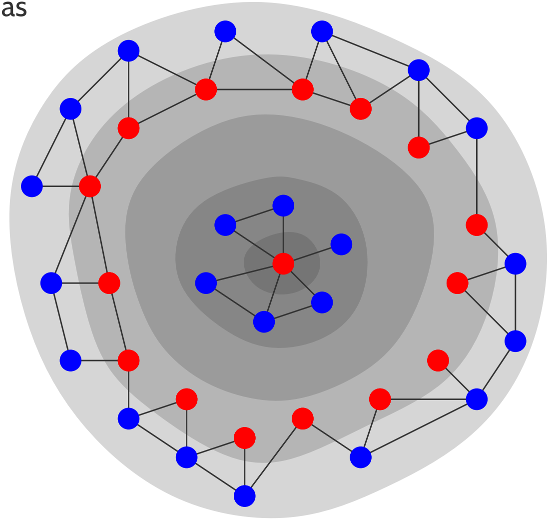
- color the graph as before with $p = |H| + 1$ colors.
- If H is induced subgraph then it is contained in a subset of $|H|$ colors.



Baker's Technique

We want to know whether G has H as an induced subgraph.

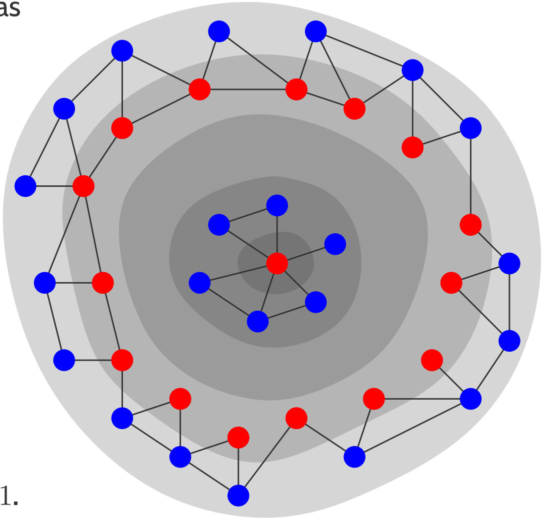
- color the graph as before with $p = |H| + 1$ colors.
- If H is induced subgraph then it is contained in a subset of $|H|$ colors.
- Enumerate all subsets of $|H|$ colors and search for H.



Baker's Technique

We want to know whether G has H as an induced subgraph.

- color the graph as before with $p = |H| + 1$ colors.
- If H is induced subgraph then it is contained in a subset of $|H|$ colors.
- Enumerate all subsets of $|H|$ colors and search for H.
- Use Courcelle on induced graph of treewith $\leq 3p + 1$.

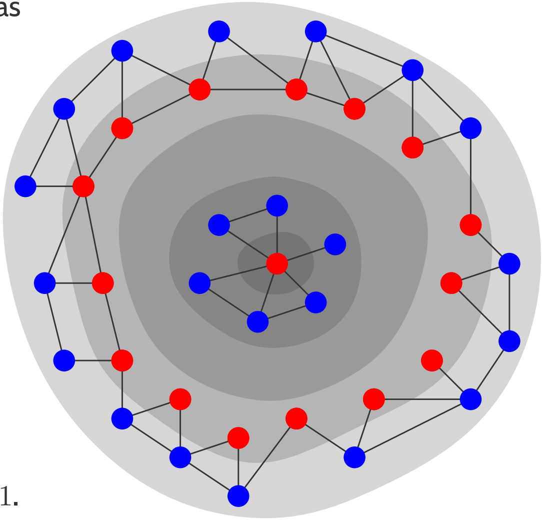


Baker's Technique

We want to know whether G has H as an induced subgraph.

- color the graph as before with $p = |H| + 1$ colors.
- If H is induced subgraph then it is contained in a subset of $|H|$ colors.
- Enumerate all subsets of $|H|$ colors and search for H.
- Use Courcelle on induced graph of treewith $\leq 3p + 1$.

Run time $\binom{p}{p-1} \cdot f(3p+1, |H|) \cdot n$.



We used the following observation of planar graphs.

- For every p one can color the graph with $p+1$ colors such that every set of p colors induces a graph with treewith at most $3p+1$.

Low Treedepth Colorings

We used the following observation of planar graphs.

- For every p one can color the graph with $p+1$ colors such that every set of p colors induces a graph with treewith at most $3p+1$.

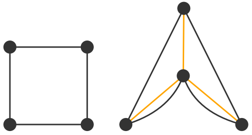
We can get something similar for bounded expansion.

- For every p one can color the graph with $f(p)$ colors such that every set of p colors induces a graph with treedepth at most p.

Treedepth

Definition

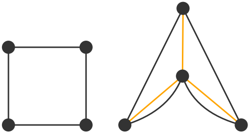
The *treedepth* of a graph G is the minimum height of a rooted forest on $V(G)$ such that all edges of G go between ancestors and descendants.

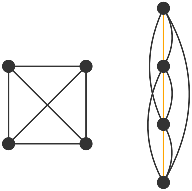


Treedepth

Definition

The *treedepth* of a graph G is the minimum height of a rooted forest on $V(G)$ such that all edges of G go between ancestors and descendants.

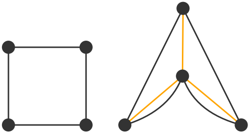


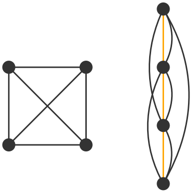


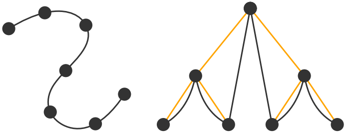
Treedepth

Definition

The *treedepth* of a graph G is the minimum height of a rooted forest on $V(G)$ such that all edges of G go between ancestors and descendants.







Treedepth vs. Treewidth

> The treewidth of a graph is as most as large as the treedepth.

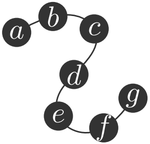
This means graph classes with bounded treewidth are more general than those with bounded treedepth.

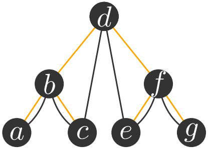
Treedepth vs. Treewidth

The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.



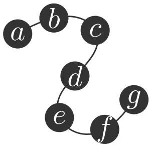


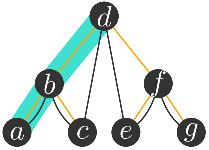
Treedepth vs. Treewidth

The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.



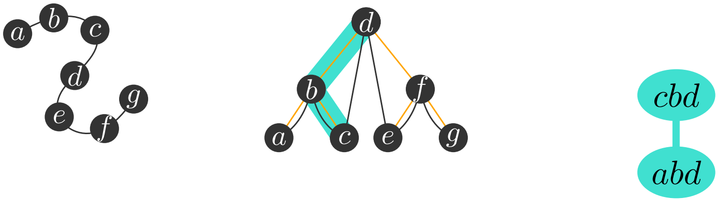


Treedepth vs. Treewidth

The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.

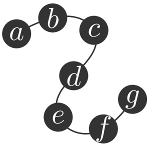


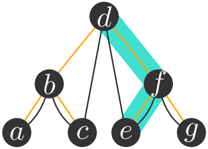
Treedepth vs. Treewidth

The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.



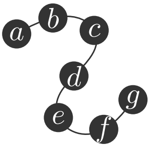


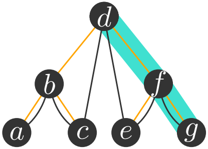
Treedepth vs. Treewidth

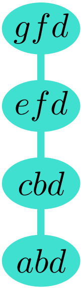
The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.





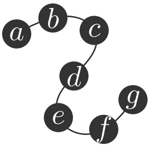


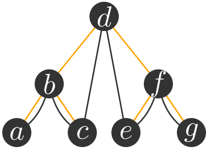
Treedepth vs. Treewidth

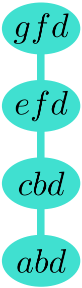
The treewidth of a graph is as most as large as the treedepth.

This means graph classes with bounded treewidth are more general than those with bounded treedepth.

Proof: The set of all paths from leafs to the root yield a tree decomposition.

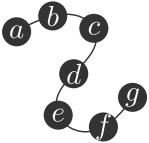


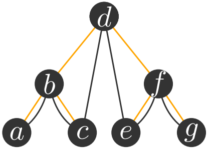




Treedepth of Paths

A treedepth of a path with n vertices is exactly $\lceil \log(n+1) \rceil$.





Low Treedepth Colorings

Nešetřil, Ossona de Mendez

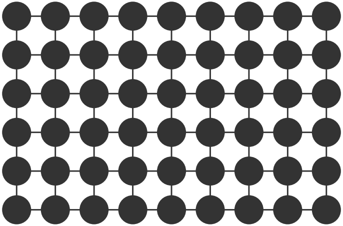
A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph?

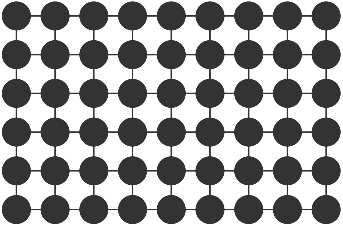


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?

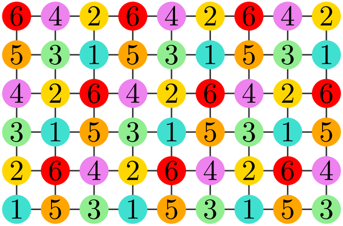


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?

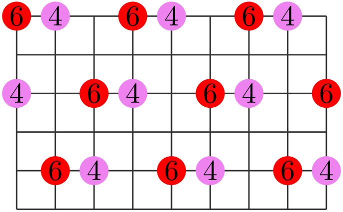


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?

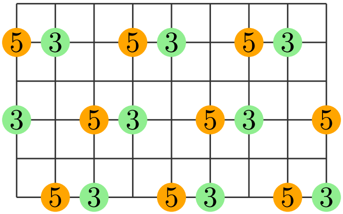


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?

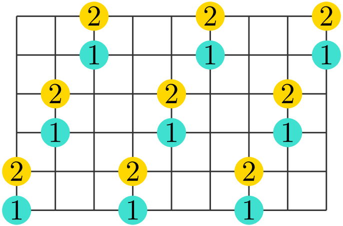


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?

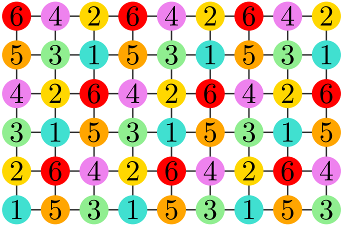


Low Treedepth Colorings

Nešetřil, Ossona de Mendez

A graph class $\mathcal{C}$ has bounded expansion iff there exists a function f such that for every $G \in \mathcal{C}$ and $p \in \mathbb{N}$ one can color G with $f(p)$ colors and every set of p colors induces a graph with treedepth $\leq p$.

What is $f(2)$ for this graph? How many colors do we need here such that every set of 2 colors has treedepth ≤ 2?



Low Treedepth Colorings

How many colors do we need to color a tree such that every set of p colors induces a graph with treedepth at most p?

Low Treedepth Colorings

How many colors do we need to color a tree such that every set of p colors induces a graph with treedepth at most p?

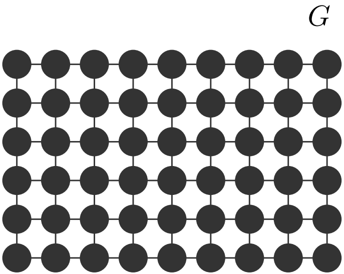
Color it with $p+1$ colors slicewise.

Existential Model-Checking

We can now use low-treedepth colorings to prove fpt existential model-checking.

Existential Model-Checking

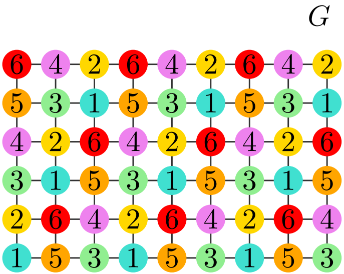
Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

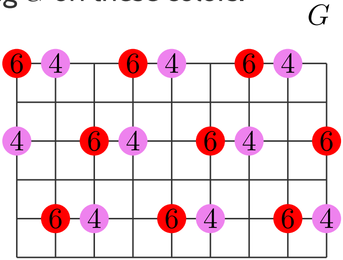
- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

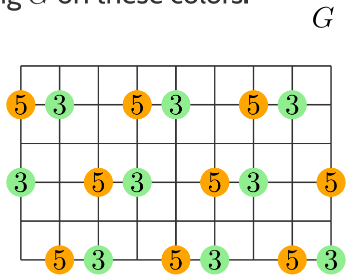
- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

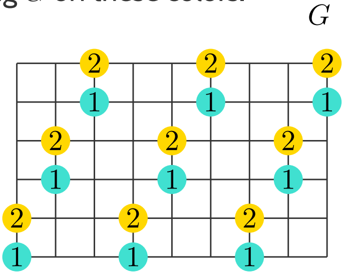
- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

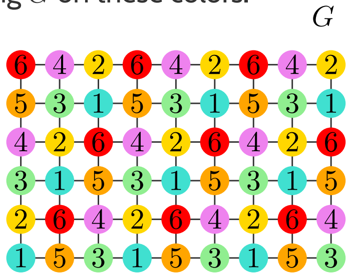
- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

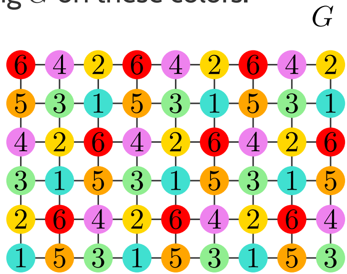
- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.
- We consider $\binom{f(|H|)}{|H|}$ color sets and for each we search for H in time $g(|H|) \cdot n$ using Courcelle.

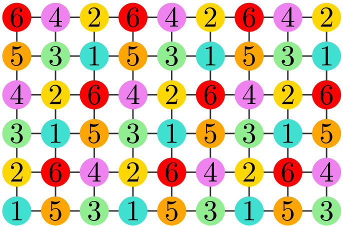


Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.
- We consider $\binom{f(|H|)}{|H|}$ color sets and for each we search for H in time $g(|H|) \cdot n$ using Courcelle.
- Total run time $\binom{f(|H|)}{|H|} g(|H|) \cdot n$.

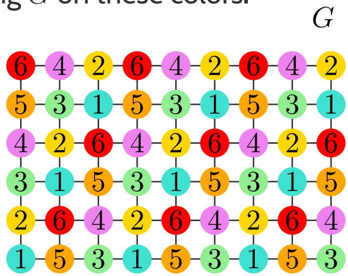
G



Existential Model-Checking

Let $\mathcal{C}$ be a class with bounded expansion, having low treedepth colorings with function $f(p)$. We want to know in time $h(|H|) \cdot n$ whether a graph $G \in \mathcal{C}$ has H as induced subgraph.

- For $p = |H|$ compute low treedepth coloring of G with $f(p)$ colors.
- H occurs in G iff there exists a set of $|H|$ colors such that H occurs in graph obtained by inducing G on these colors.
- We consider $\binom{f(|H|)}{|H|}$ color sets and for each we search for H in time $g(|H|) \cdot n$ using Courcelle.
- Total run time $\binom{f(|H|)}{|H|} g(|H|) \cdot n$.
- Plus time needed to compute coloring!



Main Result for Sparse Graphs

After we previously proved the result for existential model-checking, we now prove the full version.

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 φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

This is the most involved proof presented in this course. We will again use low-treedepth colorings.

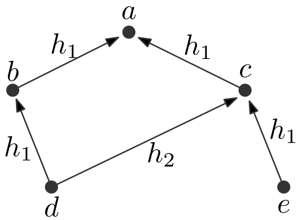
Functional Graphs

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

- $h_i : V \to V$ are unary functions
- $R_i \subseteq V$ are unary relations
- $Q_i \in \{0, 1\}$ are nullary relations

$h_1(d) = b$

$h_2(d) = c$

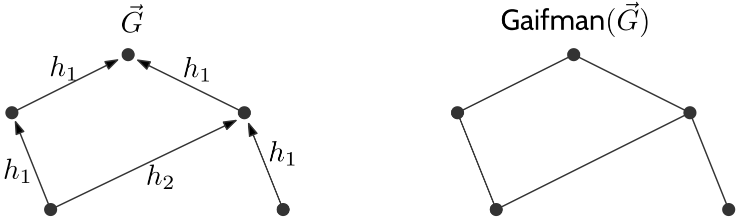


You can think of it as follows

- $h_i(u) = v$ equals directed edge from u to v of the ith type,
- $R_i(u)$ equals labeling u with ith label.
- Q_i equals a globally acessible truth value.

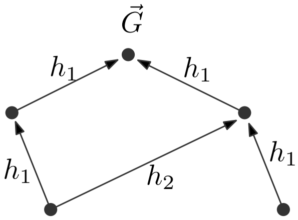
Classes

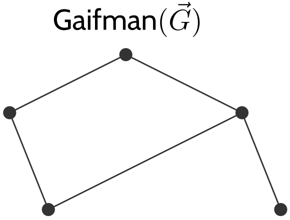
For a functional graph $\vec{G}$, the graph Gaifman($\vec{G}$) has the same vertex set and edges uv iff $h_i(u) = v$ or $h_i(v) = u$ for some i.



Classes

For a functional graph $\vec{G}$, the graph 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 *bounded expansion* if the class of their Gaifman graphs has.

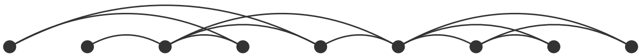
Reduction

If we can solve the model-checking problem in time $f(|\varphi|)n$ on functional graph classes with bounded expansion then we can also do it in time $f(|\varphi|)n$ on normal graph classes with bounded expansion.

Reduction

If we can solve the model-checking problem in time $f(|\varphi|)n$ on functional graph classes with bounded expansion then we can also do it in time $f(|\varphi|)n$ on normal graph classes with bounded expansion.

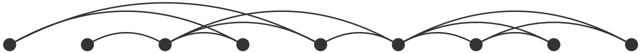
Proof: Let $\mathcal{C}$ be a normal class with bounded expansion. Then $\mathcal{C}$ has bounded degeneracy d.



Reduction

If we can solve the model-checking problem in time $f(|\varphi|)n$ on functional graph classes with bounded expansion then we can also do it in time $f(|\varphi|)n$ on normal graph classes with bounded expansion.

Proof: Let $\mathcal{C}$ be a normal class with bounded expansion. Then $\mathcal{C}$ has bounded degeneracy d.



Compute degeneracy ordering. Let $h_i(v)$ point to ith left neighbor of v. Replace in φ every occurence of $x{\sim}y$ with

$$\bigwedge_{i=1}^{d} h_i(x) = y \vee h_i(y) = x.$$

Main Result for Functional Structures

We therefore want to prove the following.

Theorem (Dvořák, Král, Thomas 2013)

Let $\vec{\mathcal{C}}$ be a functional graph class with bounded expansion. For every graph $\vec{G} \in \vec{\mathcal{C}}$ and FO formula φ one can decide whether $\vec{G} \models \varphi$ in time $f(|\varphi|)n$.

Quantifier Elimination

Convert formula to prenex normal form.

$$\vec{G} \models \exists x_1 \left(\qquad\qquad\qquad\qquad \right)$$

Quantifier Elimination

Convert formula to prenex normal form.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\qquad \qquad \qquad \right) \right)$$

Quantifier Elimination

Convert formula to prenex normal form.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \qquad\qquad \right)\right)$$

Quantifier Elimination

Convert formula to prenex normal form.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}} \right) \right)$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\underbrace{\exists x_3 \; \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}}}_{\text{replace with quantifier-free}} \right) \right)$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}' \models \exists x_1 \left(\forall x_2 \ \overbrace{\varphi'(x_1 x_2)}^{\text{quantifier-free}} \right)$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}' \models \exists x_1 \left(\underbrace{\forall x_2 \quad \overbrace{\varphi'(x_1 x_2)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}} \right)$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}'' \models \exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}'' \models \underbrace{\exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}}$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}''' \models \overbrace{\varphi'''}^{\text{quantifier-free}}$$

Quantifier Elimination

Convert formula to prenex normal form. Gradually simplify formula by removing quantifiers one by one from the inside.

$$\vec{G}''' \models \overbrace{\varphi'''}^{\text{quantifier-free}}$$

We shift complexity from the formula to the graph. G', G'', G''' have same vertices and edges but additional unary and nullary relations.

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ \exists y\, \varphi(y\bar{x}) & \longrightarrow & \varphi'(\bar{x}) \end{array}$$

Quantifier Elimination

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ \exists y\, \varphi(y\bar{x}) & \longrightarrow & \varphi'(\bar{x}) \end{array}$$

Should still be short. Length depends only on φ.

Quantifier Elimination

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

$$\vec{G} \qquad \longrightarrow \qquad \vec{G}'$$

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

Should still be short. Length depends only on φ.

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

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

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

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

Quantifier Elimination

We say there is a **quantifier elimination procedure** for a class $\mathcal{C}$ if the following holds.

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

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

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result.

$$\vec{G} \models \exists x_1 \left(\qquad\qquad\qquad\qquad \right)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\qquad\qquad\qquad \right) \right)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \qquad\qquad \right)\right)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\exists x_3 \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}} \right) \right)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result.

$$\vec{G} \models \exists x_1 \left(\forall x_2 \left(\underbrace{\exists x_3 \overbrace{\varphi(x_1 x_2 x_3)}^{\text{quantifer-free}}}_{\text{replace with quantifier-free}} \right) \right)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}' \models \exists x_1 \left(\forall x_2 \quad \overbrace{\varphi'(x_1 x_2)}^{\text{quantifier-free}} \right)$$

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}' \models \exists x_1 \neg \Big(\exists x_2 \overbrace{\neg \varphi'(x_1 x_2)}^{\text{quantifier-free}} \Big)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}' \models \exists x_1 \neg \Big(\underbrace{\exists x_2 \overbrace{\neg \varphi'(x_1 x_2)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}} \Big)$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}'' \models \exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}'' \models \underbrace{\exists x_1 \quad \overbrace{\varphi''(x_1)}^{\text{quantifier-free}}}_{\text{replace with quantifier-free}}$$

Quantifier Elimination

If there is a quantifier elimination procedure for classes with bounded expansion, this proves our main result. We use De Morgan to get rid of universal quantifiers.

$$\vec{G}''' \models \overbrace{\varphi'''}^{\text{quantifier-free}}$$

Roadmap

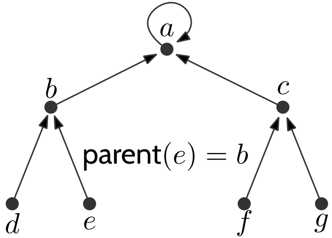
Roadmap: Construct quantifier elimination for graph classes of increasing complexity.

- forests of bounded depth
- bounded treedepth
- bounded expansion

Functional Forests

The simplest functional structures we work with are *functional forests*.

There are unary relations (labels) and exactly one unary function "parent" describing the parent relation of a rooted forest (roots point to themselves).



Functional Forests

Let $\mathcal{C}$ be a class of functional forests with bounded depth. Then $\mathcal{C}$ has a quantifier elimination procedure.

$$\vec{G} \qquad \qquad \vec{G}'$$

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

Functional Forests

Since $\varphi(y\bar{x})$ is quantifier free, it is a boolean combination of atoms of the form

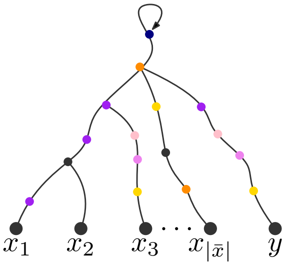
- R for some nullary relation R
- $R(\text{parent}^i(s))$ for unary R, $i \leq d$ and variable $s \in y\bar{x}$
- $\text{parent}^i(s) = \text{parent}^j(t)$ for $i, j \leq d$ and variables $s, t \in y\bar{x}$

Functional Forests

Since $\varphi(y\bar{x})$ is quantifier free, it is a boolean combination of atoms of the form

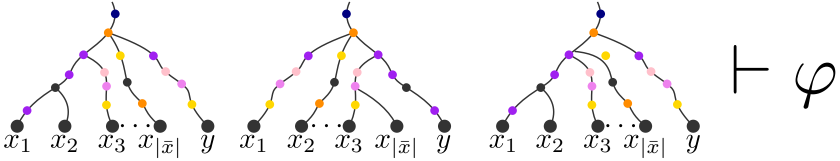
- R for some nullary relation R
- $R(\text{parent}^i(s))$ for unary R, $i \leq d$ and variable $s \in y\bar{x}$
- $\text{parent}^i(s) = \text{parent}^j(t)$ for $i, j \leq d$ and variables $s, t \in y\bar{x}$

$\varphi(y\bar{x})$ can only talk about the connections between and labelings of ancestors of $y\bar{x}$.



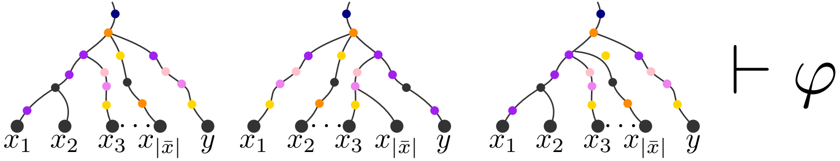
Functional Forests

Let $\mathbb{T}$ be the set of all such trees that imply φ.



Functional Forests

Let $\mathbb{T}$ be the set of all such trees that imply φ.

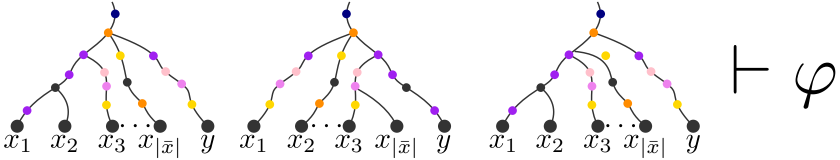


We see every $T \in \mathbb{T}$ as a formula $T(y\bar{x})$ checking if ancestor tree of $y\bar{x}$ equals T. Then

$$\exists y\, \varphi(y\bar{x}) = \exists y \bigvee_{T \in \mathbb{T}} T(y\bar{x})$$

Functional Forests

Let $\mathbb{T}$ be the set of all such trees that imply φ.



We see every $T \in \mathbb{T}$ as a formula $T(y\bar{x})$ checking if ancestor tree of $y\bar{x}$ equals T. Then

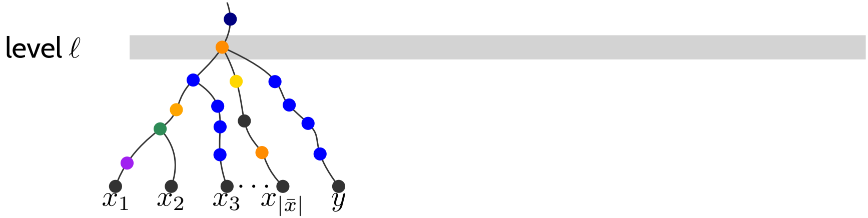
$$\exists y\, \varphi(y\bar{x}) = \exists y \bigvee_{T \in \mathbb{T}} T(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$

Functional Forests

We focus on a single subformula $\exists y \, T(y\bar{x})$.

Functional Forests

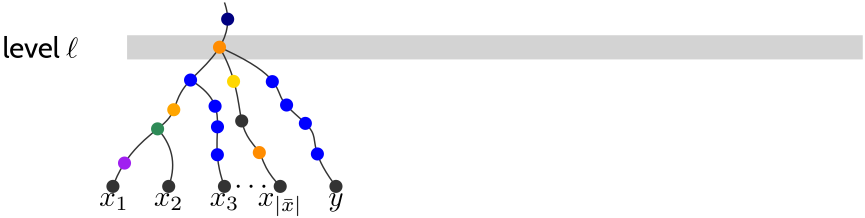
We focus on a single subformula $\exists y\, T(y\bar{x})$.



Functional Forests

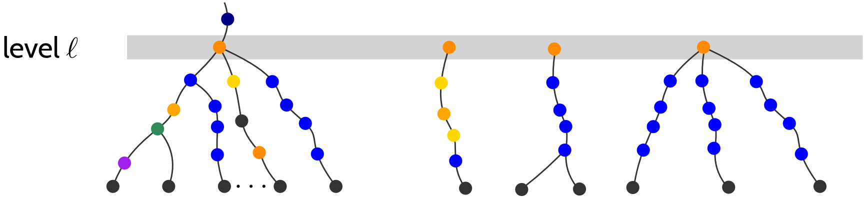
We focus on a single subformula $\exists y \, T(y\bar{x})$.

It enforces that the ancestors of $\bar{x}$ induce a certain tree and that there exists y that hits this tree via a "special path" with certain labels at a certain ancestor at height l.



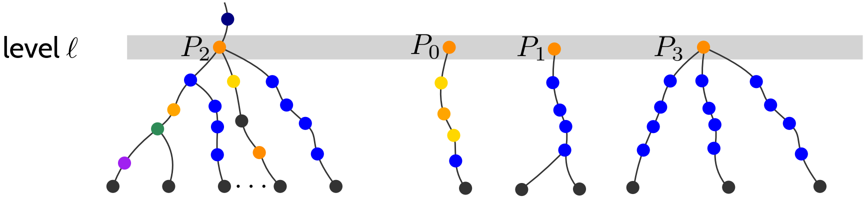
Functional Forests

We now augment the graph.



Functional Forests

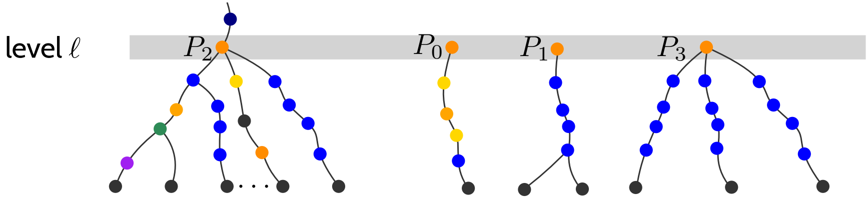
We now augment the graph. For all v at level l use label $P_i(v)$ to store that i children of v can be completed to form a special path.



Functional Forests

We now augment the graph. For all v at level l use label $P_i(v)$ to store that i children of v can be completed to form a special path.

- round all numbers larger than $|\bar{x}|$ up to ∞.

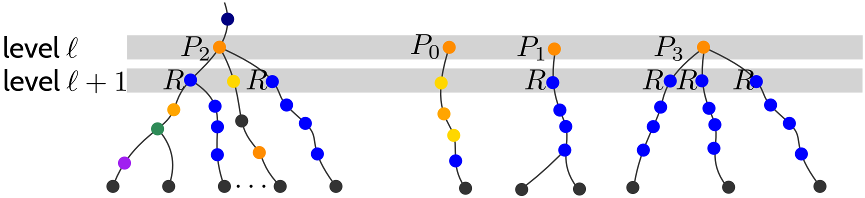


Functional Forests

We now augment the graph. For all v at level l use label $P_i(v)$ to store that i children of v can be completed to form a special path.

- round all numbers larger than $|\bar{x}|$ up to ∞.

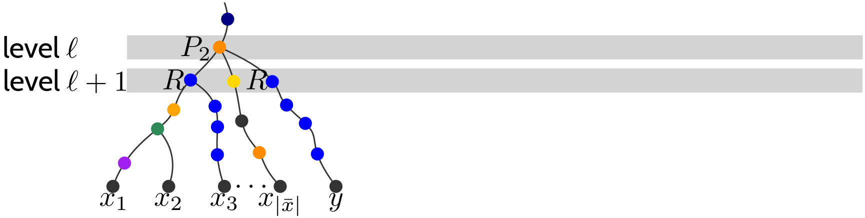
For all w at level $l+1$ add label $R(w)$ if w lies on a special path.



Functional Forests

We construct a quantifier-free formula $T'(\bar{x})$ equivalent to $\exists y\, T(y\bar{x})$. This formula can only check the ancestor tree of $\bar{x}$.

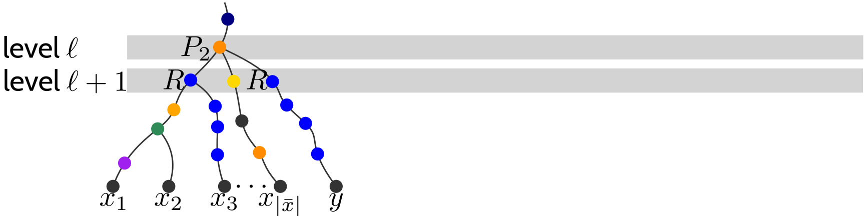
We construct a quantifier-free formula $T'(\bar{x})$ equivalent to $\exists y\, T(y\bar{x})$. This formula can only check the ancestor tree of $\bar{x}$.



Functional Forests

We construct a quantifier-free formula $T'(\bar{x})$ equivalent to $\exists y\, T(y\bar{x})$. This formula can only check the ancestor tree of $\bar{x}$.

If we want to know if there is y reaching an ancestor s of $\bar{x}$, we can check whether s has more ancestors than are in the tree of $\bar{x}$.



Functional Forests

For every ancestor tree $T \in \mathbb{T}$ we can perform quantifier elimination.

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ \exists y\, T(y\bar{x}) & \to & T'(\bar{x}) \end{array}$$

Functional Forests

For every ancestor tree $T \in \mathbb{T}$ we can perform quantifier elimination.

$$\begin{array}{ccc} \vec{G} & & \vec{G}' \\ \exists y\, T(y\bar{x}) & \rightarrow & T'(\bar{x}) \end{array}$$

$\vec{G} \models \exists y\, T(y\bar{v})$ iff $\vec{G}' \models T'(\bar{v})$ for all $\bar{v} \in V(\vec{G})^{|\bar{x}|}$.

However, we want quantifier elimination for our original formula

$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$

However, we want quantifier elimination for our original formula

$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$

$$\vec{G}$$

$$\exists y\, T_1(y\bar{x}) \vee \exists y\, T_2(y\bar{x})$$

However, we want quantifier elimination for our original formula

$$\exists y \, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y \, T(y\bar{x}).$$

$$\vec{G}$$
$$\exists y \, T_1(y\bar{x})$$

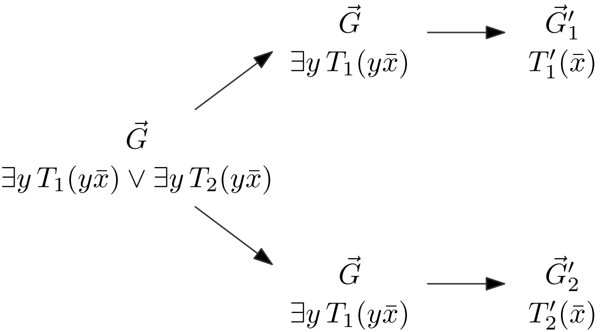
$$\vec{G}$$
$$\exists y \, T_1(y\bar{x}) \vee \exists y \, T_2(y\bar{x})$$

$$\vec{G}$$
$$\exists y \, T_1(y\bar{x})$$

Functional Forests

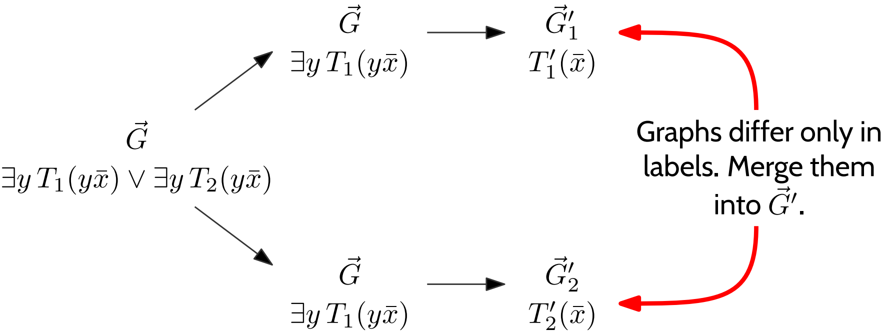
However, we want quantifier elimination for our original formula

$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$



However, we want quantifier elimination for our original formula

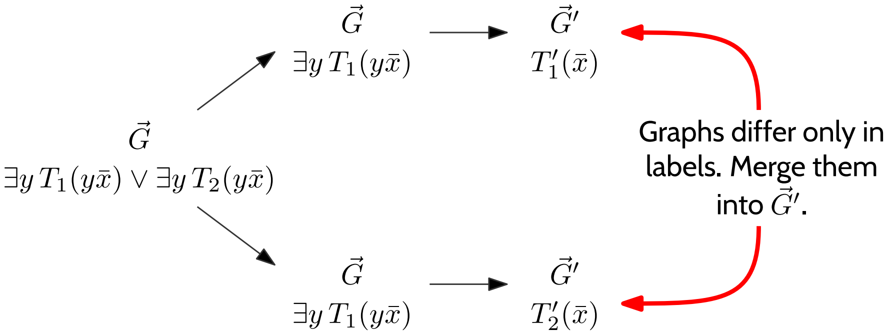
$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$



Functional Forests

However, we want quantifier elimination for our original formula

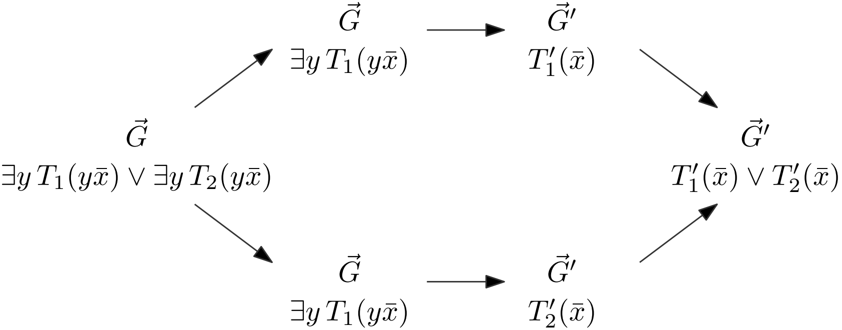
$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$



Functional Forests

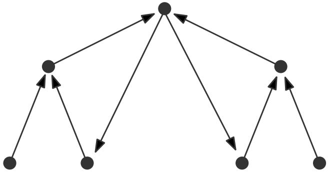
However, we want quantifier elimination for our original formula

$$\exists y\, \varphi(y\bar{x}) = \bigvee_{T \in \mathbb{T}} \exists y\, T(y\bar{x}).$$



Bounded Treedepth

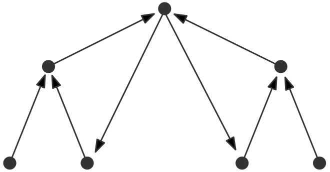
Next, we consider functional graph classes whose Gaifman graphs have *bounded treedepth*.



Bounded Treedepth

Next, we consider functional graph classes whose Gaifman graphs have *bounded treedepth*.

Let $\mathcal{C}$ be a class whose Gaifman graphs have *bounded treedepth*. Then $\mathcal{C}$ has a quantifier elimination procedure.

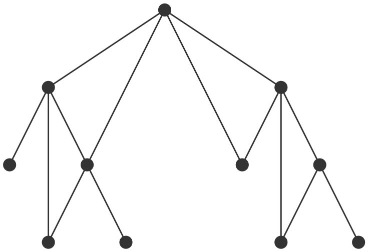


Bounded Treedepth

bounded
treedepth

$$\vec{G}$$

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

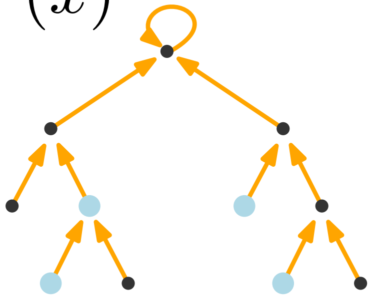


bounded treedepth

forest of bounded depth

$$\vec{G} \longrightarrow \vec{G}'$$

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



Bounded Treedepth

bounded treedepth

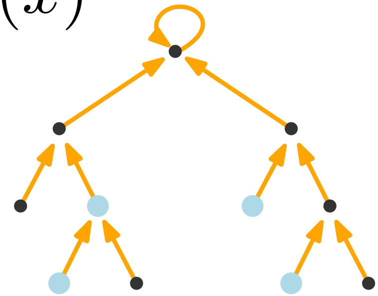
$$\vec{G}$$

forest of bounded depth

$$\vec{G}''$$

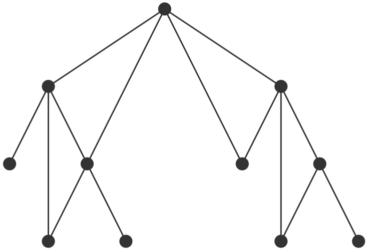
$$\exists y\, \varphi(y\bar{x}) \quad \longrightarrow \quad \varphi''(\bar{x})$$

Use previous theorem for quantifier elimination on forests.



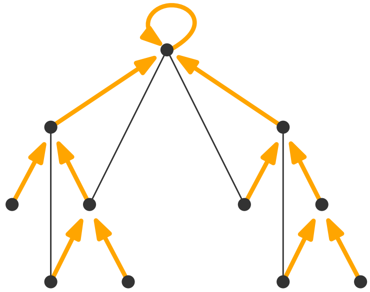
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$.



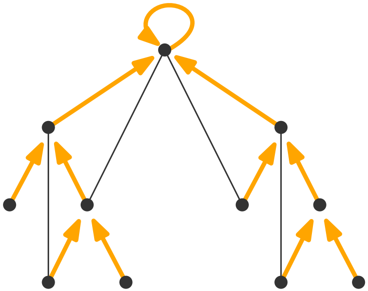
Bounded Treedepth

- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional.



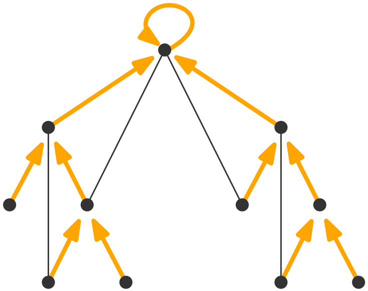
Bounded Treedepth

- 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.



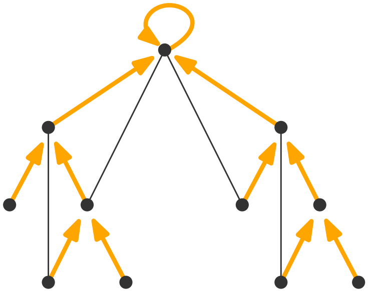
Bounded Treedepth

- 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.



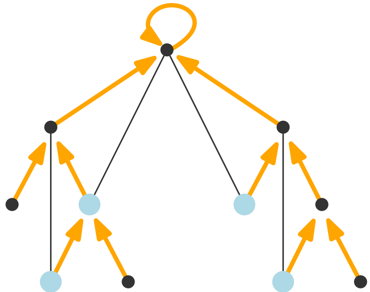
Bounded Treedepth

- 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.



Bounded Treedepth

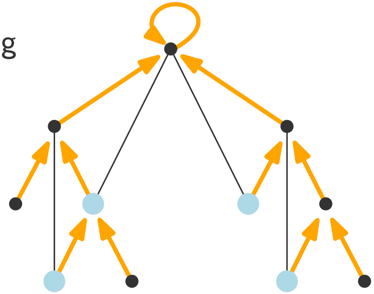
- Consider Gaifman graph of $\vec{G}$. Build depth-first search tree and make it functional. Graphs with treedepth d contain no paths longer than 2^d. We have a functional forest of depth at most 2^d.
- All edges go between ancestors in the tree. Encode edges of $\vec{G}$ by predicates at lower endpoints.



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.

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

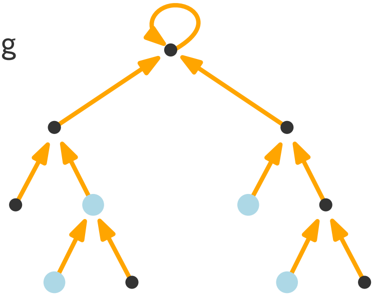


Bounded Treedepth

- 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.

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

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

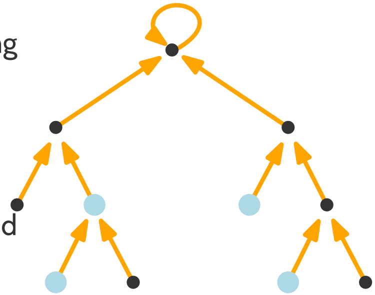


Bounded Treedepth

- 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.

  $P_{i,j,\updownarrow}(v)$: "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.

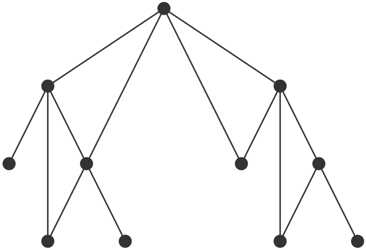


Bounded Treedepth

bounded
treedepth

$$\vec{G}$$

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



Bounded Treedepth

bounded treedepth

$$\vec{G}$$

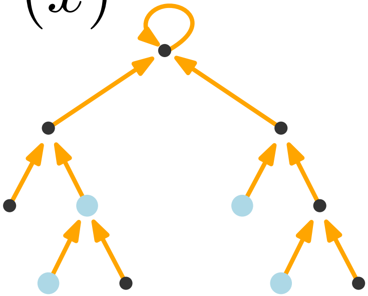
$$\exists y\, \varphi(y\bar{x})$$

$$\longrightarrow$$

forest of bounded depth

$$\vec{G}'$$

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



Bounded Treedepth

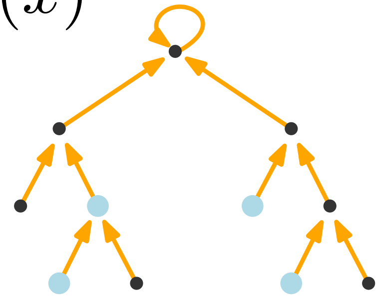
bounded treedepth → $\vec{G}$

forest of bounded depth → $\vec{G}''$

$$\vec{G} \quad \longrightarrow \quad \vec{G}''$$

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

Use previous theorem for quantifier elimination on forests.



Bounded Expansion

The last step.

Let $\mathcal{C}$ be a class with bounded expansion. Then $\mathcal{C}$ has a quantifier elimination procedure.

Bounded Expansion

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

Bounded Expansion

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

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists y\, x_i \underset{g}{\rightsquigarrow} y \wedge y \underset{p}{\rightsquigarrow} x_j$.

Bounded Expansion

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

Replace all function applications such as $p(g(x_i)) = x_j$ with directed labeled edges such as $\exists y \, x_i \underset{g}{\rightsquigarrow} y \wedge y \underset{p}{\rightsquigarrow} x_j$.

The result is a formula $\exists \bar{y} \, \psi(\bar{y}\bar{x})$ on a normal (non-functional) directed edge-labeled graph G.

Bounded Expansion

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

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

Bounded Expansion

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

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

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $|\bar{y}\bar{x}|$ colors has treedepth $\leq g(|\bar{y}\bar{x}|)$.

Bounded Expansion

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

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

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S| = |\bar{y}\bar{x}|}} \text{colors}(\bar{v}) \subseteq S \wedge$$

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $|\bar{y}\bar{x}|$ colors has treedepth $\leq g(|\bar{y}\bar{x}|)$.

Bounded Expansion

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

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

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

Let Λ be the colors of a low-treedepth coloring of G where every subgraph on $|\bar{y}\bar{x}|$ colors has treedepth $\leq g(|\bar{y}\bar{x}|)$.

Bounded Expansion

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

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

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

For all graphs $G[S]$ (of bounded treedepth) we perform quantifier elimination.

Bounded Expansion

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

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

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

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S| = |\bar{y}\bar{x}|}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}'_S \models \psi_S(\bar{v}) \quad \text{iff}$$

For all graphs $G[S]$ (of bounded treedepth) we perform quantifier elimination.

Bounded Expansion

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

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

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

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S| = |\bar{y}\bar{x}|}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}' \models \psi_S(\bar{v}) \quad \text{iff}$$

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

Bounded Expansion

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

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

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

$$\bigvee_{\substack{S \subseteq \Lambda \\ |S| = |\bar{y}\bar{x}|}} \text{colors}(\bar{v}) \subseteq S \wedge \vec{G}' \models \psi_S(\bar{v}) \quad \text{iff}$$

$$\vec{G}' \models \bigvee_{\substack{S \subseteq \Lambda \\ |S| = |\bar{y}\bar{x}|}} \text{colors}(\bar{v}) \subseteq S \wedge \psi_S(\bar{v}).$$

Finally, pull out the graph.

Summary

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.