Definability equals recognizability for graphs of bounded treewidth

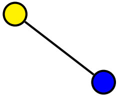
Mikołaj Bojańczyk, Michał Pilipczuk

Institute of Informatics, University of Warsaw

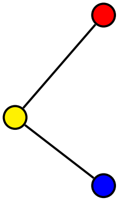
Warwick Workshop on Algorithms, Logic and Structure
December 12^{th}, 2016

How to construct graphs

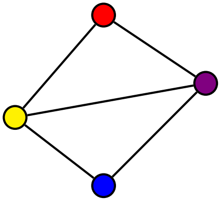
How to construct graphs



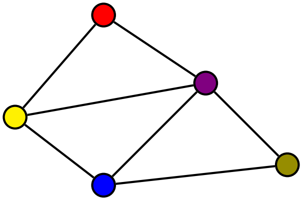
How to construct graphs



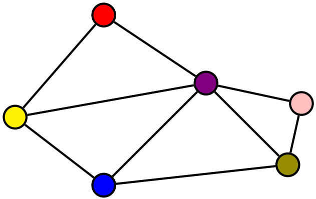
How to construct graphs



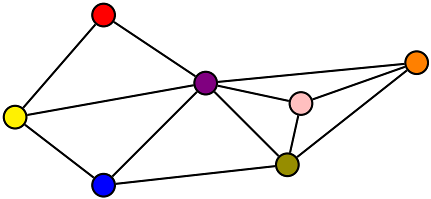
How to construct graphs



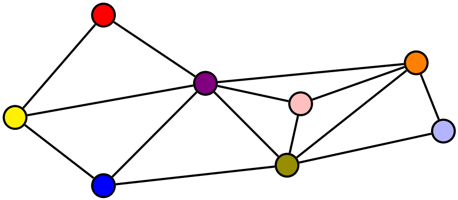
How to construct graphs



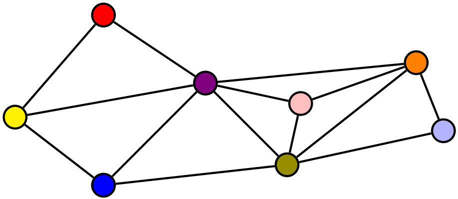
How to construct graphs



How to construct graphs



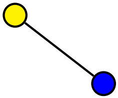
How to construct graphs



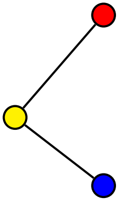
Idea: Keep only a small number of vertices in memory.

How to construct graphs

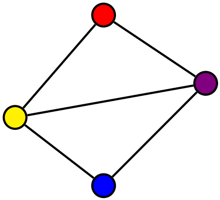
How to construct graphs



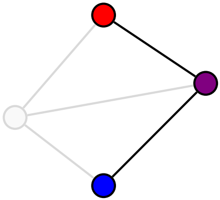
How to construct graphs



How to construct graphs

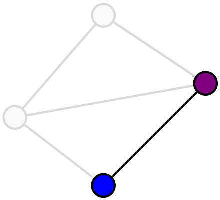


How to construct graphs



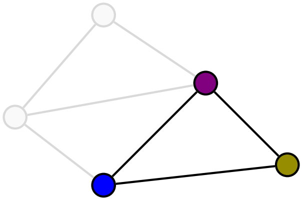
Forget a present active vertex

How to construct graphs



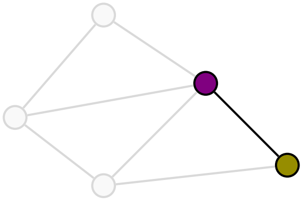
Forget a present active vertex

How to construct graphs



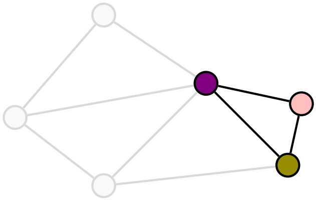
Introduce a new active vertex

How to construct graphs



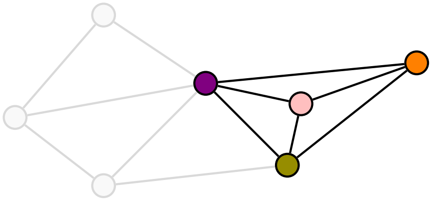
Forget

How to construct graphs



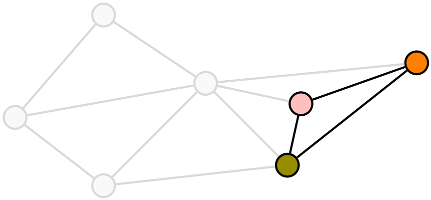
Introduce

How to construct graphs



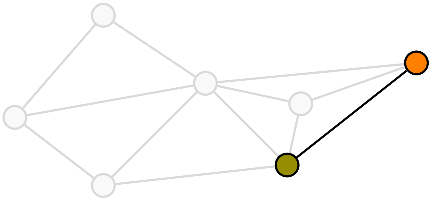
Introduce

How to construct graphs

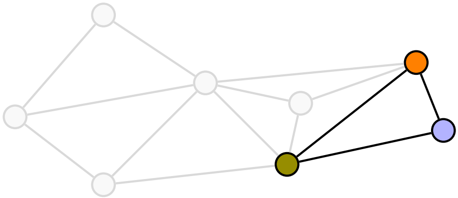


Forget

How to construct graphs

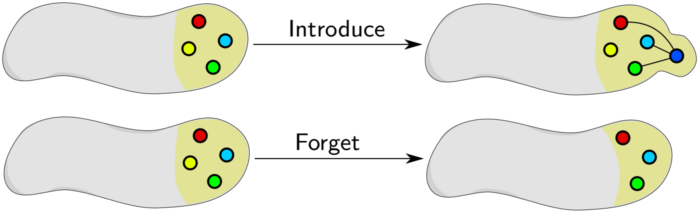


Forget

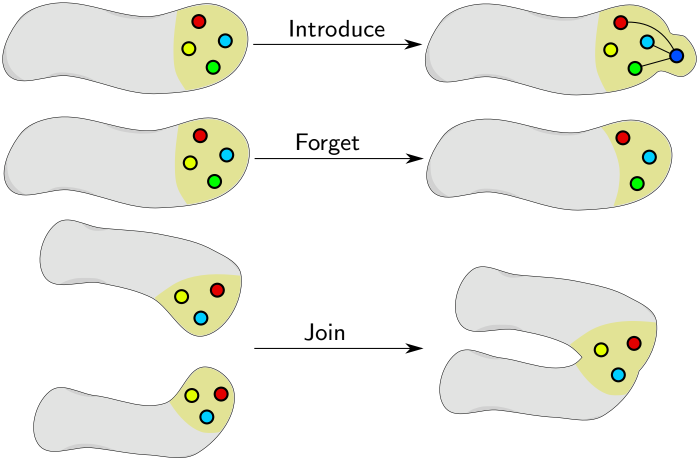


Introduce

Operations



Operations



Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.

Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.

Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.
- **Pathwidth** of a graph G:
  the minimum k needed to construct G using **introduce** and **forget**.

Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.
- **Pathwidth** of a graph G:
  the minimum k needed to construct G using **introduce** and **forget**.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.

Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.
- **Pathwidth** of a graph G:
  the minimum k needed to construct G using **introduce** and **forget**.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.
  - With each node associate its *bag*: the vertices active at the moment.

Interface graph algebra

- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.
- **Pathwidth** of a graph G:
  the minimum k needed to construct G using **introduce** and **forget**.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.
  - With each node associate its *bag*: the vertices active at the moment.
  - The parameter k is the *width* of the decomposition.

MSO on graphs

- **Monadic Second Order** logic on graphs:

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

Courcelle's theorem

Π expressible in MSO $\Rightarrow$
Π can be verified in linear time on graphs of constant treewidth.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

Courcelle's theorem

Π expressible in MSO $\Rightarrow$
Π can be verified in linear time on graphs of constant treewidth.

- **Proof**:

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

Courcelle's theorem

Π expressible in MSO ⇒
Π can be verified in linear time on graphs of constant treewidth.

- **Proof**:
  - Transform a formula φ expressing Π on a graph into an equivalent formula ψ on a labeled tree encoding the tree decomposition.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

Courcelle's theorem

Π expressible in MSO $\Rightarrow$
Π can be verified in linear time on graphs of constant treewidth.

- **Proof**:
  - Transform a formula φ expressing Π on a graph into an equivalent formula ψ on a labeled tree encoding the tree decomposition.
  - Transform ψ into an equivalent automaton $\mathcal{A}_\psi$ and run it on the decomposition.

MSO on graphs

- **Monadic Second Order** logic on graphs:
  - Language for expressing graph properties.
  - We can quantify existentially/universally over vertices, edges, vertex subsets, edge subsets.
  - We can check incidence, belonging, etc.
  - **Example 1**: 3-Colorability
  - **Example 2**: Hamiltonicity

Courcelle's theorem

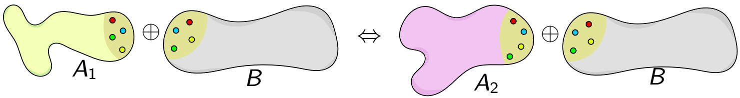
Π expressible in MSO $\Rightarrow$
Π can be verified in linear time on graphs of constant treewidth.

- **Proof**:
  - Transform a formula φ expressing Π on a graph into an equivalent formula ψ on a labeled tree encoding the tree decomposition.
  - Transform ψ into an equivalent automaton $\mathcal{A}_\psi$ and run it on the decomposition.
- **Courcelle's conjecture**: If Π can be verified by an automaton on a tree decomposition, then Π is expressible in MSO.

Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \quad\Leftrightarrow\quad A_1 \oplus B \in \Pi \text{ iff } A_2 \oplus B \in \Pi$$
$$\text{for every } k\text{-interface graph } B.$$

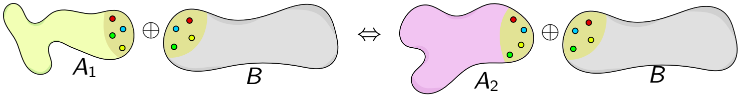


Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \qquad \Leftrightarrow \qquad A_1 \oplus B \in \Pi \ \text{ iff } \ A_2 \oplus B \in \Pi$$

$$\text{for every } k\text{-interface graph } B.$$

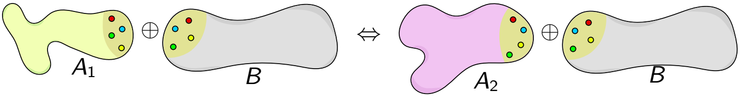


- Π is **recognizable** if it is k-recognizable for every k.

Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \qquad \Leftrightarrow \qquad A_1 \oplus B \in \Pi \text{ iff } A_2 \oplus B \in \Pi$$
$$\text{for every } k\text{-interface graph } B.$$

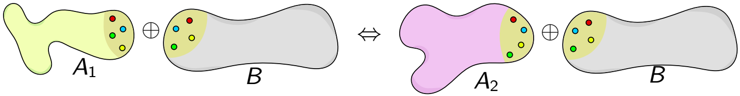


- Π is **recognizable** if it is k-recognizable for every k.
- **Idea**: Recognizable properties can be verified using tree automata working on tree decompositions.

Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \qquad \Leftrightarrow \qquad A_1 \oplus B \in \Pi \ \text{ iff } \ A_2 \oplus B \in \Pi$$
$$\text{for every } k\text{-interface graph } B.$$

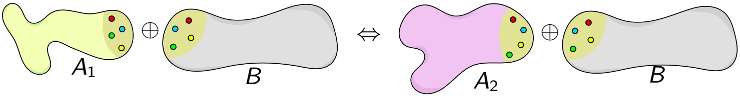


- Π is **recognizable** if it is k-recognizable for every k.
- **Idea**: Recognizable properties can be verified using tree automata working on tree decompositions.
- **Fact**: Every MSO-definable graph property is recognizable.

Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \qquad \Leftrightarrow \qquad A_1 \oplus B \in \Pi \text{ iff } A_2 \oplus B \in \Pi$$
$$\text{for every } k\text{-interface graph } B.$$

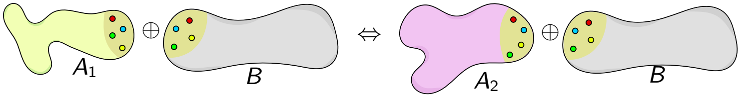


- Π is **recognizable** if it is k-recognizable for every k.
- **Idea**: Recognizable properties can be verified using tree automata working on tree decompositions.
- **Fact**: Every MSO-definable graph property is recognizable.
- **Converse**: Is every recognizable graph property MSO-definable?

Recognizability

- Graph property Π is k-**recognizable** if the following Myhill-Nerode relation $\equiv_k$ over k-interface graphs has finite index.

$$A_1 \equiv_k A_2 \qquad \Leftrightarrow \qquad A_1 \oplus B \in \Pi \;\text{ iff }\; A_2 \oplus B \in \Pi$$
$$\text{for every } k\text{-interface graph } B.$$



- Π is **recognizable** if it is k-recognizable for every k.
- **Idea**: Recognizable properties can be verified using tree automata working on tree decompositions.
- **Fact**: Every MSO-definable graph property is recognizable.
- **Converse**: Is every recognizable graph property MSO-definable?
  - **WRONG** for multiple reasons.

Courcelle's conjecture

Courcelle's conjecture Courcelle; ∼'90

Suppose

- Π is a recognizable graph property, and
- $\mathcal{T}_k$ is the class of graphs of treewidth at most k, for some constant k.

Then $\Pi \cap \mathcal{T}_k$ can be defined in MSO with modular counting predicates.

Courcelle's conjecture

Courcelle's conjecture — Courcelle; $\sim$'90

Suppose

- Π is a recognizable graph property, and
- $\mathcal{T}_k$ is the class of graphs of treewidth at most k, for some constant k.

Then $\Pi \cap \mathcal{T}_k$ can be defined in MSO with modular counting predicates.

Theorem — Bojańczyk, P.; 2016

Courcelle's conjecture holds.

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.
- Take a tree decomposition of the given graph G.

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.
- Take a tree decomposition of the given graph G.
- Guess existentially the run of $\mathcal{A}$ on the tree decomposition.

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.
- Take a tree decomposition of the given graph G.
- Guess existentially the run of $\mathcal{A}$ on the tree decomposition.
- Verify that it is correct and that it accepts.

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.
- **Take** a tree decomposition of the given graph G.
- Guess existentially the run of $\mathcal{A}$ on the tree decomposition.
- Verify that it is correct and that it accepts.
- **Caveat**: We are given **only** a graph,
  not a graph together with its tree decomposition!

Attempt on the proof

- By the finiteness of the Myhill-Nerode equivalence relation, there is a tree automaton $\mathcal{A}$ that:
  - Works on tree decompositions of width k.
  - Recognizes exactly tree decompositions of graphs from Π.
- **Take** a tree decomposition of the given graph G.
- Guess existentially the run of $\mathcal{A}$ on the tree decomposition.
- Verify that it is correct and that it accepts.
- **Caveat**: We are given **only** a graph,
  not a graph together with its tree decomposition!
- Everything boils down to "defining" in MSO some tree decomposition of bounded width.

MSO transductions

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

MSO transductions

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

- MSO **transduction**: a formal way of describing nondeterministic "MSO-definable" transformations of relational structures.

MSO transductions

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

- MSO **transduction**: a formal way of describing nondeterministic "MSO-definable" transformations of relational structures.
- One can existentially guess some sets, and then interpret the structure of the decomposition using MSO predicates.

MSO transductions

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

- MSO **transduction**: a formal way of describing nondeterministic "MSO-definable" transformations of relational structures.
- One can existentially guess some sets, and then interpret the structure of the decomposition using MSO predicates.
  - **Example**: Guess a subset of **red** edges, and for each vertex u create a bag consisting of all vertices reachable from u via **red** edges.

MSO transductions

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

- MSO **transduction**: a formal way of describing nondeterministic "MSO-definable" transformations of relational structures.
- One can existentially guess some sets, and then interpret the structure of the decomposition using MSO predicates.
  - **Example**: Guess a subset of **red** edges, and for each vertex u create a bag consisting of all vertices reachable from u via **red** edges.
- **Fact**: If a property is MSO-definable after the intepretation, then it is also MSO-definable before.

Main theorem

There is a (nondeterministic) MSO transduction that, given a graph of treewidth k, outputs its tree decomposition of width at most $f(k)$, for some function f.

- MSO **transduction**: a formal way of describing nondeterministic "MSO-definable" transformations of relational structures.
- One can existentially guess some sets, and then interpret the structure of the decomposition using MSO predicates.
  - **Example**: Guess a subset of **red** edges, and for each vertex u create a bag consisting of all vertices reachable from u via **red** edges.
- **Fact**: If a property is MSO-definable after the intepretation, then it is also MSO-definable before.
- **Now**: A combinatorial notion of an "MSO-definable" decomposition.

Guidance system

Guidance system

A *guidance system* Λ in a graph G is a set of rooted forests

$$(F_1, F_2, \ldots, F_k)$$

where $V(F_i) = V(G)$ and $F_i \subseteq G$ for each i.

Guidance system

A *guidance system* Λ in a graph G is a set of rooted forests

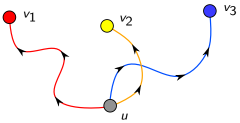
$$(F_1, F_2, \dots, F_k)$$

where $V(F_i) = V(G)$ and $F_i \subseteq G$ for each i.

- For each $u \in V(G)$, define k-tuple $\Lambda(u)$ as

$$\Lambda(u) = (v_1, v_2, \dots, v_k),$$

where v_i is the root of the tree of F_i that contains u.



Guidance system

Guidance system

A *guidance system* Λ in a graph G is a set of rooted forests

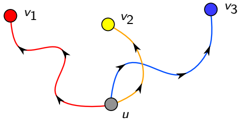
$$(F_1, F_2, \ldots, F_k)$$

where $V(F_i) = V(G)$ and $F_i \subseteq G$ for each i.

- For each $u \in V(G)$, define k-tuple $\Lambda(u)$ as

$$\Lambda(u) = (v_1, v_2, \ldots, v_k),$$

where v_i is the root of the tree of F_i that contains u.

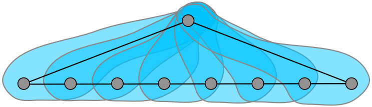


- A vertex subset X is *captured* by Λ if $X \subseteq \Lambda(u)$ for some vertex u.

Capturing tree decompositions

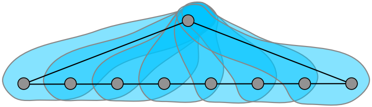
- A tree decomposition is *captured* by Λ if each of its bags is captured.

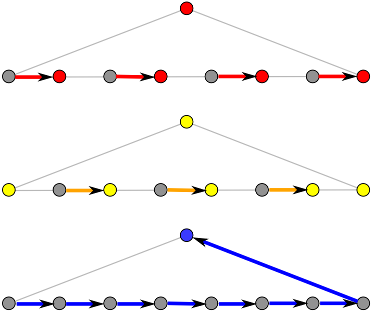
- A tree decomposition is *captured* by Λ if each of its bags is captured.



Capturing tree decompositions

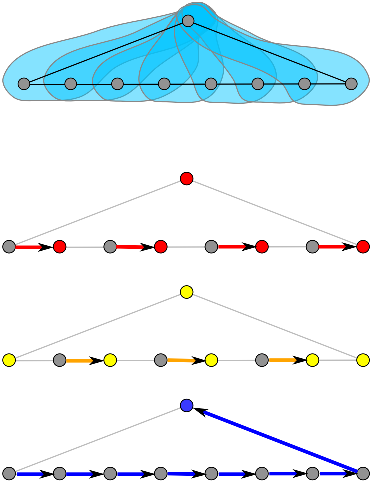
- A tree decomposition is *captured* by Λ if each of its bags is captured.

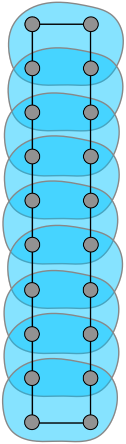




Capturing tree decompositions

- A tree decomposition is *captured* by Λ if each of its bags is captured.





Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Theorem

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{pw}(G))$ for every graph G.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Theorem

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{pw}(G))$ for every graph G.

- We would be done if Conjecture was proved.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Theorem

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{pw}(G))$ for every graph G.

- We would be done if Conjecture was proved.
- In our proof, we circumvent proving the Conjecture.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Theorem

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{pw}(G))$ for every graph G.

- We would be done if Conjecture was proved.
- In our proof, we circumvent proving the Conjecture.
- **Rest of the talk**: Proof of the Theorem.

Guided treewidth

- **Fact**: If a decomposition is captured by a guidance system of constant size, then it can be constructed by an MSO-transduction.
- **Guided treewidth** of G, denoted $\mathbf{gtw}(G)$, is the smallest size of a guidance system that captures a tree decomposition of G.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Theorem

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{pw}(G))$ for every graph G.

- We would be done if Conjecture was proved.
- In our proof, we circumvent proving the Conjecture.
- **Rest of the talk**: Proof of the Theorem.
- **Tool**: Simon's factorization forest.

Simon's factorization forest

- Suppose S is a finite semigroup.

Simon's factorization forest

- Suppose S is a finite semigroup.
- **Setting**: We are given a long word

$$a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_{n-2} \cdot a_{n-1} \cdot a_n$$

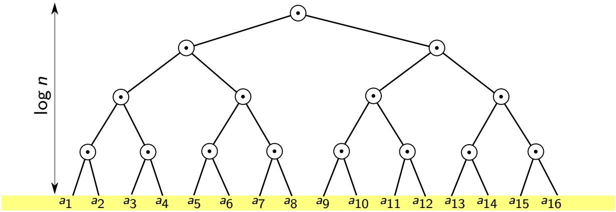
with $a_i \in S$. We want to "factorize" the product "efficiently".

- Suppose S is a finite semigroup.
- **Setting**: We are given a long word

$$a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_{n-2} \cdot a_{n-1} \cdot a_n$$

with $a_i \in S$. We want to "factorize" the product "efficiently".

- **Binary factorization**:



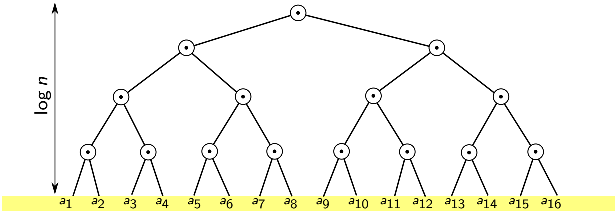
Simon's factorization forest

- Suppose S is a finite semigroup.
- **Setting**: We are given a long word

$$a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_{n-2} \cdot a_{n-1} \cdot a_n$$

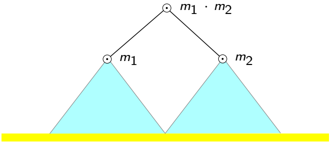
with $a_i \in S$. We want to "factorize" the product "efficiently".

- **Binary factorization**:



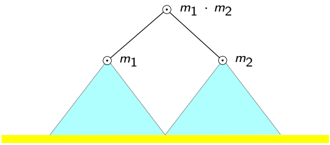
- We need **constant** depth, depending only on $|S|$.

Simon's factorization forest

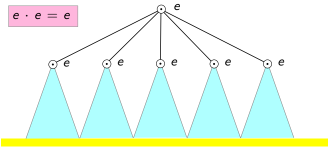


Binary node

Simon's factorization forest

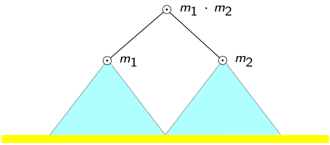


Binary node

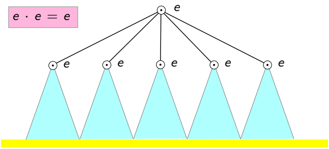


Idempotent node

Simon's factorization forest



Binary node

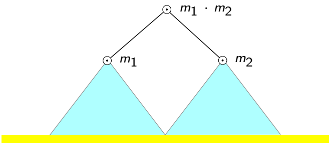


Idempotent node

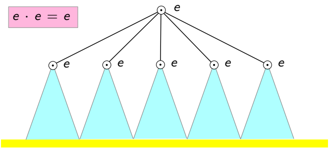
Simon's factorization forest theorem [Simon'90; Kufleitner'08]

Every word over S has a factorization of depth at most $3|S|$ that uses binary and idempotent nodes.

Simon's factorization forest



Binary node



Idempotent node

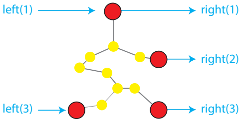
Simon's factorization forest theorem [Simon'90; Kufleitner'08]

Every word over S has a factorization of depth at most $3|S|$ that uses binary and idempotent nodes.

path decomp. of width k $\Rightarrow$ word over a semigroup of size $f(k)$

apply induction on the depth of the factorization forest

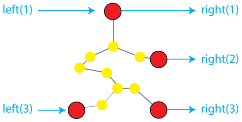
- **Bi-interface graph**:
  Graph with left and right interfaces, numbered from 1 to k.



$\mathbb{G}_1$

Bi-interface graphs

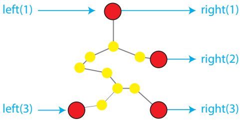
- **Bi-interface graph**:
  Graph with left and right interfaces, numbered from 1 to k.
    - Not every number has to be used.



$\mathbb{G}_1$

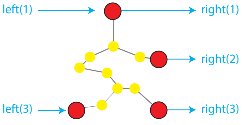
Bi-interface graphs

- **Bi-interface graph**:
  Graph with left and right interfaces, numbered from 1 to k.
  - Not every number has to be used.
  - If a vertex is both a left and a right interface, its number in both interfaces is the same.

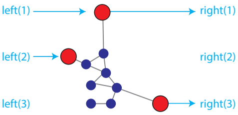


$\mathbb{G}_1$

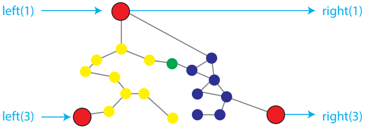
- **Bi-interface graph**:
  Graph with left and right interfaces, numbered from 1 to k.
  - Not every number has to be used.
  - If a vertex is both a left and a right interface, its number in both interfaces is the same.
  - Natural gluing operation.



$\mathbb{G}_1$



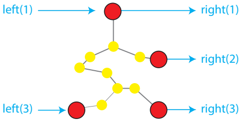
$\mathbb{G}_2$



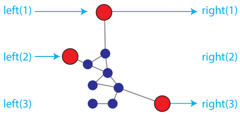
$\mathbb{G}_1 \oplus \mathbb{G}_2$

Bi-interface graphs

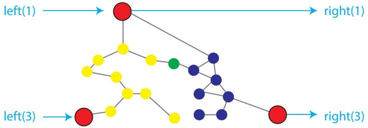
- **Bi-interface graph**:
  Graph with left and right interfaces, numbered from 1 to k.
  - Not every number has to be used.
  - If a vertex is both a left and a right interface, its number in both interfaces is the same.
  - Natural gluing operation.
  - Parameter k is the **arity** of the bi-interface graph.



$\mathbb{G}_1$



$\mathbb{G}_2$



$\mathbb{G}_1 \oplus \mathbb{G}_2$

Abstraction semigroup

- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.

Abstraction semigroup

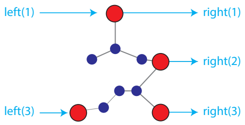
- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.
- **Lemma**: If a graph has pathwidth $\leq k$, then it can be written as $\mathbb{H}_1 \oplus \mathbb{H}_2 \oplus \ldots \oplus \mathbb{H}_t$ where $\mathbb{H}_i$ has arity k and contains no non-interface vertices (is *basic*).

Abstraction semigroup

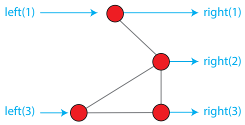
- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.
- **Lemma**: If a graph has pathwidth $\leq k$, then it can be written as $\mathbb{H}_1 \oplus \mathbb{H}_2 \oplus \ldots \oplus \mathbb{H}_t$ where $\mathbb{H}_i$ has arity k and contains no non-interface vertices (is *basic*).
- **Issue**: This semigroup is infinite.

Abstraction semigroup

- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.
- **Lemma**: If a graph has pathwidth $\leq k$, then it can be written as $\mathbb{H}_1 \oplus \mathbb{H}_2 \oplus \ldots \oplus \mathbb{H}_t$ where $\mathbb{H}_i$ has arity k and contains no non-interface vertices (is *basic*).
- **Issue**: This semigroup is infinite.
- Define **abstraction** as torso with respect to interfaces.



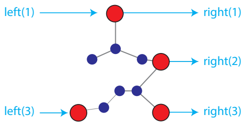
$\mathbb{G}$



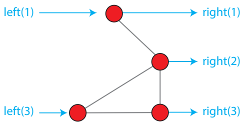
$[\![\mathbb{G}]\!]$

Abstraction semigroup

- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.
- **Lemma**: If a graph has pathwidth $\leq k$, then it can be written as $\mathbb{H}_1 \oplus \mathbb{H}_2 \oplus \ldots \oplus \mathbb{H}_t$ where $\mathbb{H}_i$ has arity k and contains no non-interface vertices (is *basic*).
- **Issue**: This semigroup is infinite.
- Define **abstraction** as torso with respect to interfaces.



$\mathbb{G}$



$[\![\mathbb{G}]\!]$

- Consider operation on basic bi-interface graphs of arity k:

$$\mathbb{G}_1 \oplus_t \mathbb{G}_2 = [\![\mathbb{G}_1 \oplus \mathbb{G}_2]\!].$$

Abstraction semigroup

- Bi-interface graphs of arity k with gluing $\oplus$ form a semigroup.
- **Lemma**: If a graph has pathwidth $\leq k$, then it can be written as $\mathbb{H}_1 \oplus \mathbb{H}_2 \oplus \ldots \oplus \mathbb{H}_t$ where $\mathbb{H}_i$ has arity k and contains no non-interface vertices (is *basic*).
- **Issue**: This semigroup is infinite.
- Define **abstraction** as torso with respect to interfaces.

- Consider operation on basic bi-interface graphs of arity k:

$$\mathbb{G}_1 \oplus_{\mathrm{t}} \mathbb{G}_2 = [\![\mathbb{G}_1 \oplus \mathbb{G}_2]\!].$$

- This forms a semigroup $\mathcal{S}$ of size $2^{\mathcal{O}(k^2)}$.

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.
- **Claim**: $\mathbf{gtw}(\mathbb{G}) \leq f(k, d)$, where d is the depth of factorization.

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.
- **Claim**: $\mathbf{gtw}(\mathbb{G}) \leq f(k, d)$, where d is the depth of factorization.
- **Goal**: Guided treewidth increases in a controlled way when gluing as in binary and idempotent nodes.

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.
- **Claim**: $\mathbf{gtw}(\mathbb{G}) \leq f(k, d)$, where d is the depth of factorization.
- **Goal**: Guided treewidth increases in a controlled way when gluing as in binary and idempotent nodes.

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.
- **Claim**: $\mathbf{gtw}(\mathbb{G}) \leq f(k, d)$, where d is the depth of factorization.
- **Goal**: Guided treewidth increases in a controlled way when gluing as in binary and idempotent nodes.

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2 + 5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$

Proof strategy

- **Idea**: Induction on the depth of Simon's factorization over $\mathcal{S}$.
- **Claim**: $\mathbf{gtw}(\mathbb{G}) \leq f(k, d)$, where d is the depth of factorization.
- **Goal**: Guided treewidth increases in a controlled way when gluing as in binary and idempotent nodes.

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2 + 5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$

- These functions stack at most $3|\mathcal{S}| = 2^{\mathcal{O}(k^2)}$ times and we are done.

Binary lemma

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

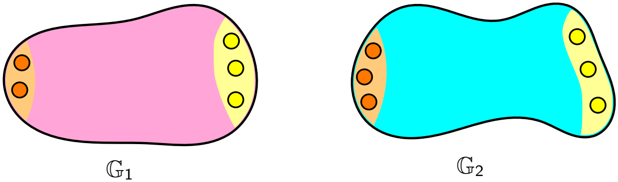
$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$

Binary lemma

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

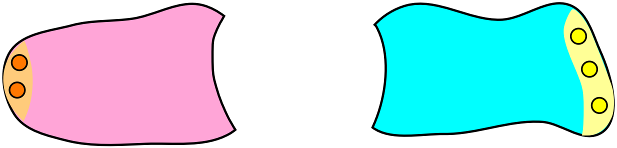
$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$



Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$



$\mathbb{G}_1$ – right $\qquad$ $\mathbb{G}_2$ – left

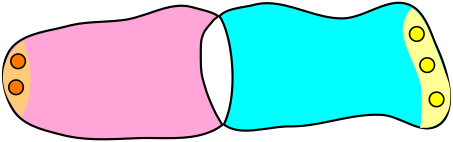
- **Fact 1**: $\mathbf{gtw}(G - u) \leq 2 \cdot \mathbf{gtw}(G)$.

Binary lemma

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$



$$(\mathbb{G}_1 - \text{right}) \uplus (\mathbb{G}_2 - \text{left})$$

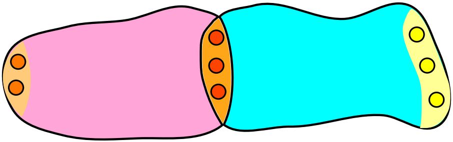
- **Fact 1**: $\mathbf{gtw}(G - u) \leq 2 \cdot \mathbf{gtw}(G)$.
- **Fact 2**: $\mathbf{gtw}(G_1 \uplus G_2) = \max(\mathbf{gtw}(G_1), \mathbf{gtw}(G_2))$.

Binary lemma

Binary lemma

If $\mathbb{G}_1$ and $\mathbb{G}_2$ are bi-interface graphs of arity k, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \mathbb{G}_2) \leq k + 2^k \cdot \max(\mathbf{gtw}(\mathbb{G}_1), \mathbf{gtw}(\mathbb{G}_2)).$$



$\mathbb{G}_1 \oplus \mathbb{G}_2$

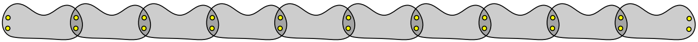
- **Fact 1**: $\mathbf{gtw}(G - u) \leq 2 \cdot \mathbf{gtw}(G)$.
- **Fact 2**: $\mathbf{gtw}(G_1 \uplus G_2) = \max(\mathbf{gtw}(G_1), \mathbf{gtw}(G_2))$.
- **Fact 3**: $\mathbf{gtw}(G) \leq \mathbf{gtw}(G - u) + 1$.

Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \le k(4k^2+5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$

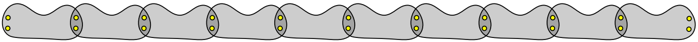


Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2+5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$



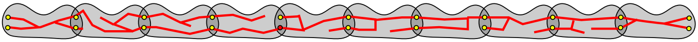
- Apply same strategy $\rightsquigarrow$ Too many interfaces to reintroduce.

Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2+5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$



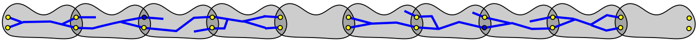
- Apply same strategy $\rightsquigarrow$ Too many interfaces to reintroduce.
- For each interface we add a spanning tree of the whole graph just to span nearby columns!

Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2+5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$



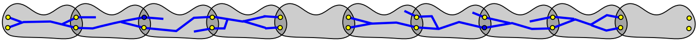
- Apply same strategy $\rightsquigarrow$ Too many interfaces to reintroduce.
- For each interface we add a spanning tree of the whole graph just to span nearby columns!
- **Solution**: Instead, span only $\mathcal{O}(k^2)$ nearby columns.

Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \ldots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \ldots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_t) \leq k(4k^2 + 5) + 8^k \cdot \max_{i=1,\ldots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$



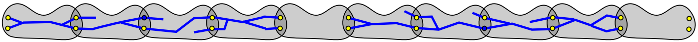
- Apply same strategy $\rightsquigarrow$ Too many interfaces to reintroduce.
- For each interface we add a spanning tree of the whole graph just to span nearby columns!
- **Solution**: Instead, span only $\mathcal{O}(k^2)$ nearby columns.
  - Here we use that abstractions are the same.

Idempotent lemma

Idempotent lemma

If $\mathbb{G}_1, \dots, \mathbb{G}_t$ are bi-int. graphs of arity k with $[\![\mathbb{G}_1]\!] = \dots = [\![\mathbb{G}_t]\!]$, then

$$\mathbf{gtw}(\mathbb{G}_1 \oplus \dots \oplus \mathbb{G}_t) \leq k(4k^2+5) + 8^k \cdot \max_{i=1,\dots,t} \{\mathbf{gtw}(\mathbb{G}_i)\}.$$



- Apply same strategy $\rightsquigarrow$ Too many interfaces to reintroduce.
- For each interface we add a spanning tree of the whole graph just to span nearby columns!
- **Solution**: Instead, span only $\mathcal{O}(k^2)$ nearby columns.
  - Here we use that abstractions are the same.
- Trees can be colored with $\mathcal{O}(k^3)$ colors and grouped into forests.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):
  - For all k, there is an MSO-transduction that given a graph of treewidth k, outputs a tree decomposition of width at most k.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):
  - For all k, there is an MSO-transduction that given a graph of treewidth k, outputs a tree decomposition of width at most k.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):
  - For all k, there is an MSO-transduction that given a graph of treewidth k, outputs a tree decomposition of width at most k.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Conjecture

There is a function f s.t. every graph of treewidth k has an optimum width tree decomposition captured by a guidance system of size $f(k)$.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):
  - For all k, there is an MSO-transduction that given a graph of treewidth k, outputs a tree decomposition of width at most k.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Conjecture

There is a function f s.t. every graph of treewidth k has an optimum width tree decomposition captured by a guidance system of size $f(k)$.

Conclusions

- **Lifting pathwidth to treewidth**:
  If $\mathbf{tw}(G) \leq k$, then there is a tree decomposition $\mathcal{T}$ of G such that
  - adhesions of $\mathcal{T}$ can be captured by a guidance system of size $f(k)$;
  - the torso of each bag has pathwidth at most $f(k)$.
  - Combine both decompositions at the level of MSO-transductions.
- **Further work** (BP; STACS 2017):
  - For all k, there is an MSO-transduction that given a graph of treewidth k, outputs a tree decomposition of width at most k.

Conjecture

There is a function f such that $\mathbf{gtw}(G) \leq f(\mathbf{tw}(G))$ for every graph G.

Conjecture

There is a function f s.t. every graph of treewidth k has an optimum width tree decomposition captured by a guidance system of size $f(k)$.

- **Thanks for attention!**