Definability equals recognizability for graphs of bounded treewidth

Mikołaj Bojańczyk, Michał Pilipczuk

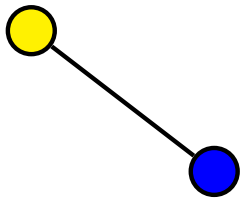
Highlights in Brussels,
September 8^{th}, 2016

Joint work with...

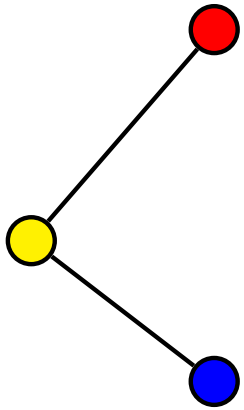
Mikołaj Bojańczyk, here receiving the **Crystal Brussels Sprout Award**.

How to construct graphs

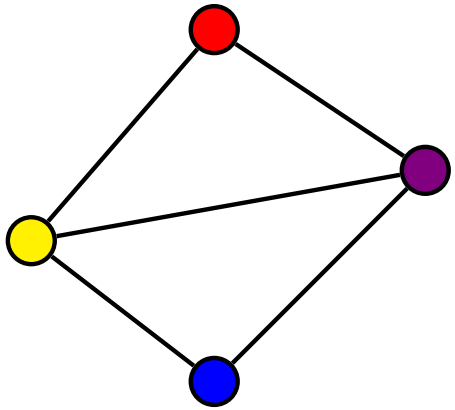
How to construct graphs



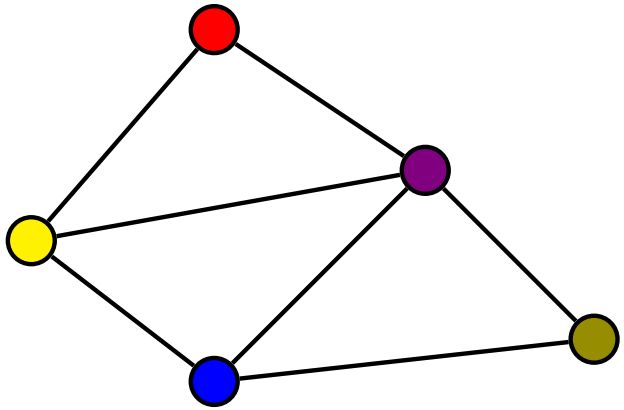
How to construct graphs



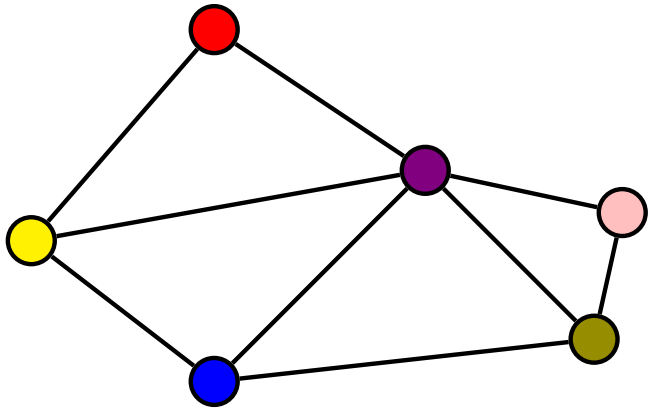
How to construct graphs



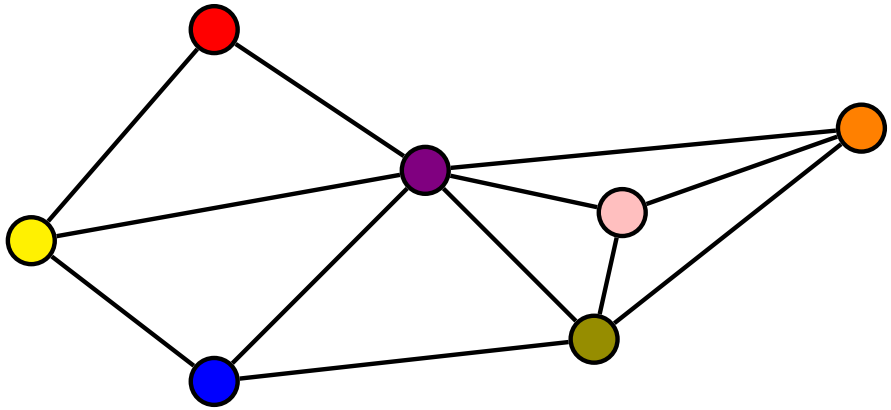
How to construct graphs



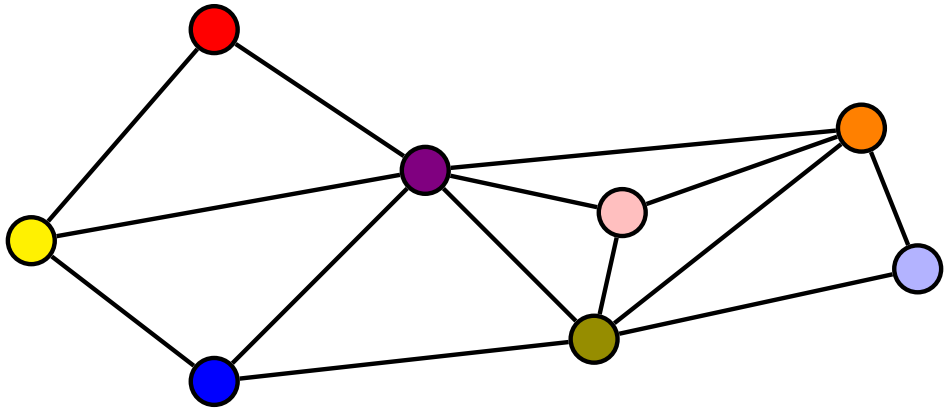
How to construct graphs



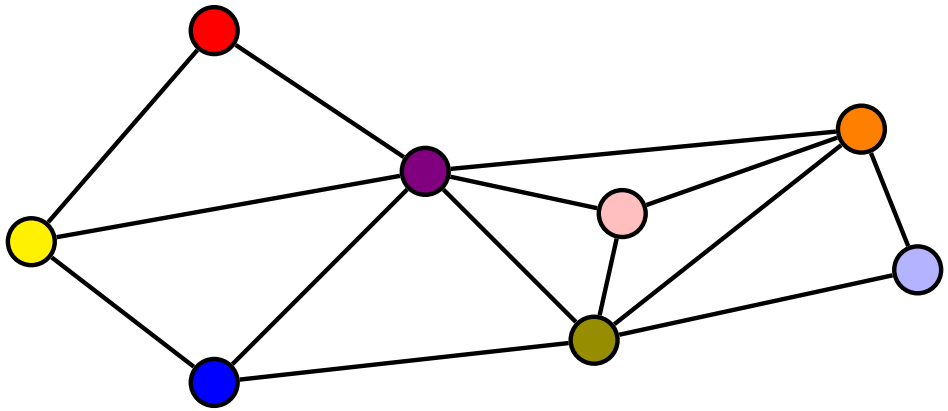
How to construct graphs



How to construct graphs



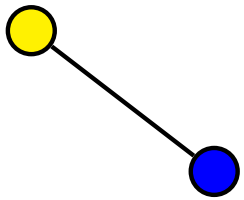
How to construct graphs



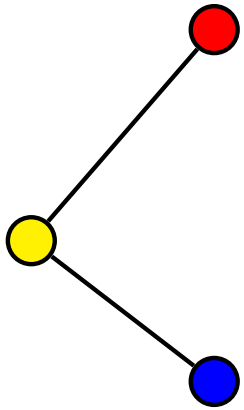
Idea: Keep only a small number of vertices in memory.

How to construct graphs

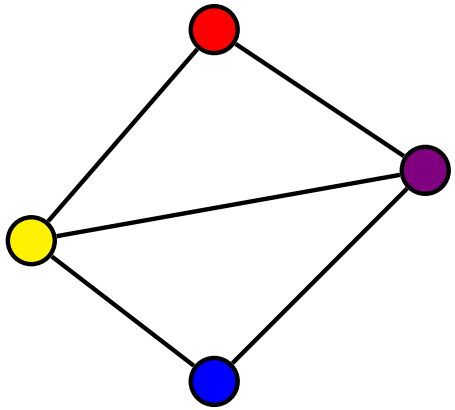
How to construct graphs



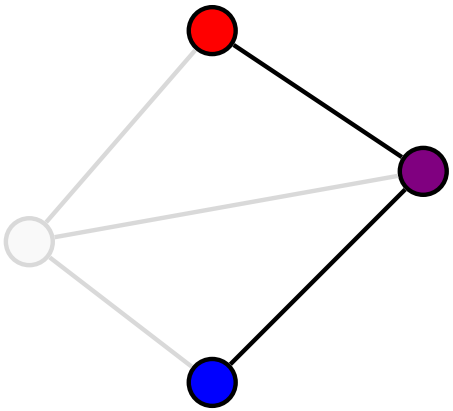
How to construct graphs



How to construct graphs

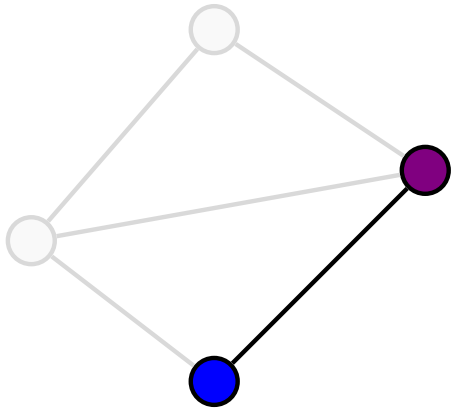


How to construct graphs



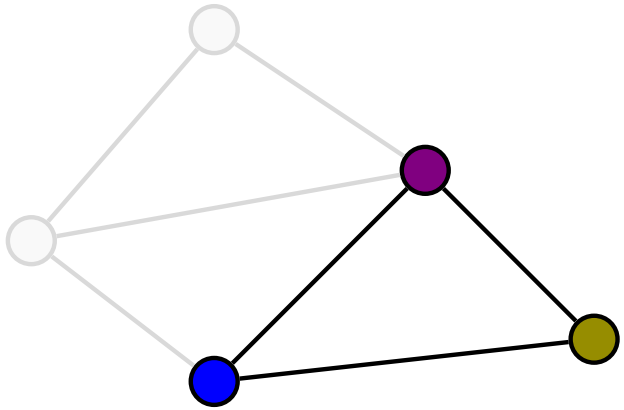
Forget a present active vertex

How to construct graphs



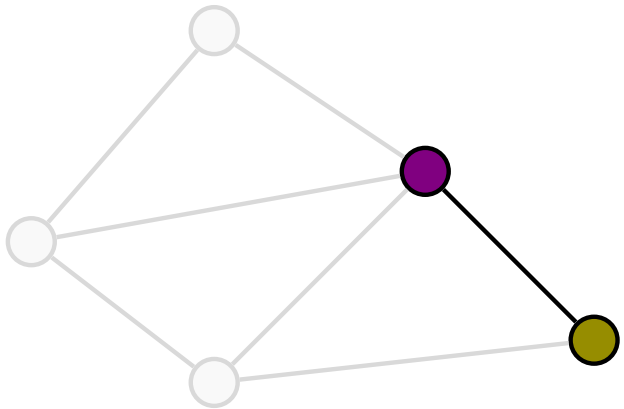
Forget a present active vertex

How to construct graphs



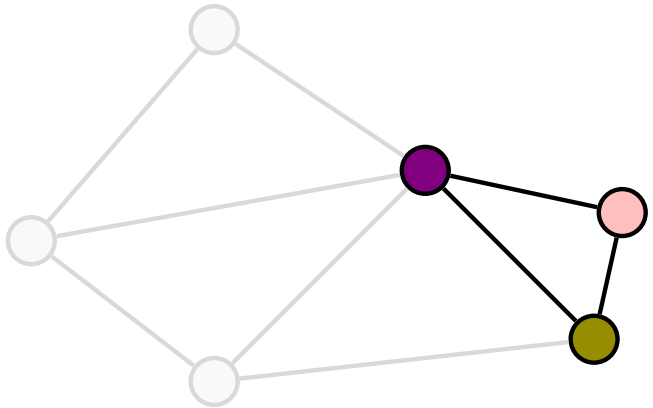
Introduce a new active vertex

How to construct graphs



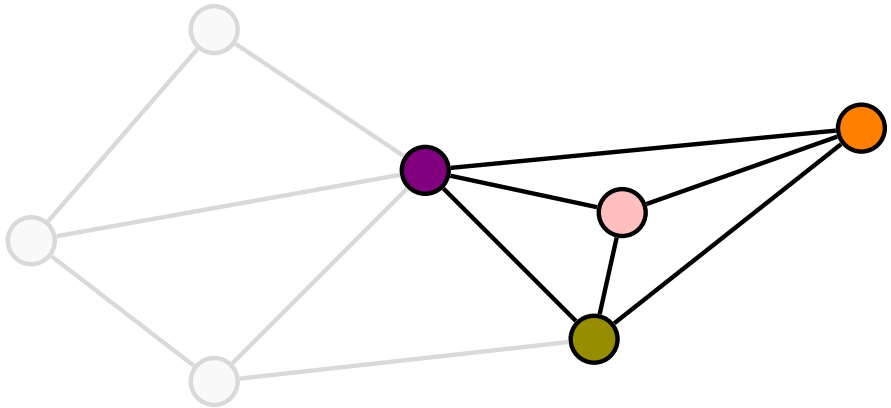
Forget

How to construct graphs



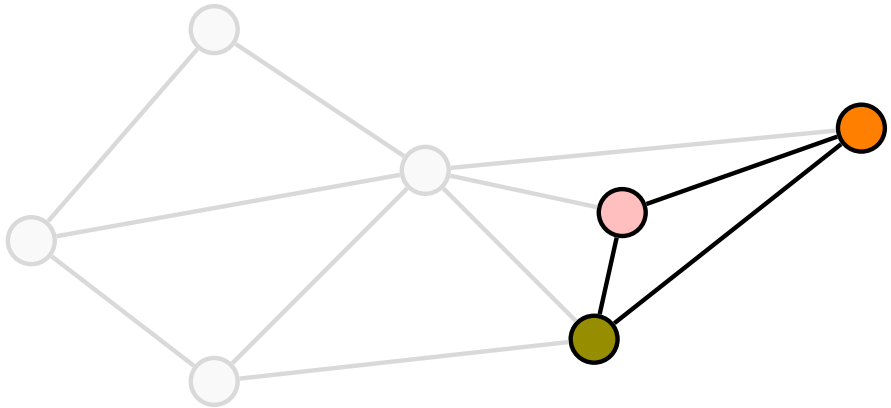
Introduce

How to construct graphs



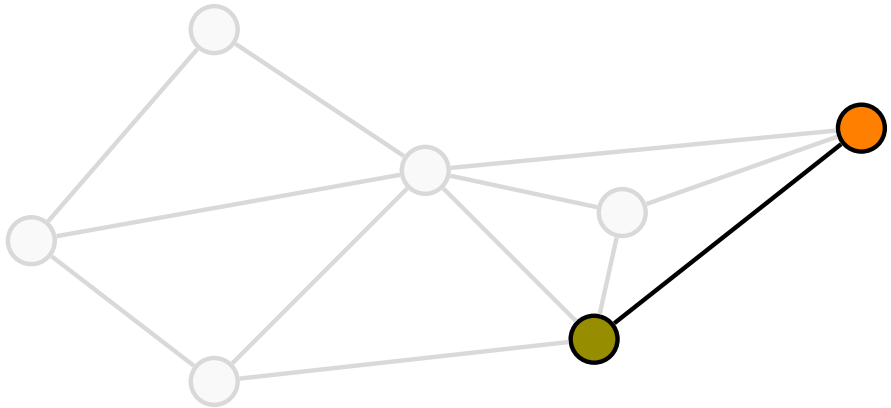
Introduce

How to construct graphs



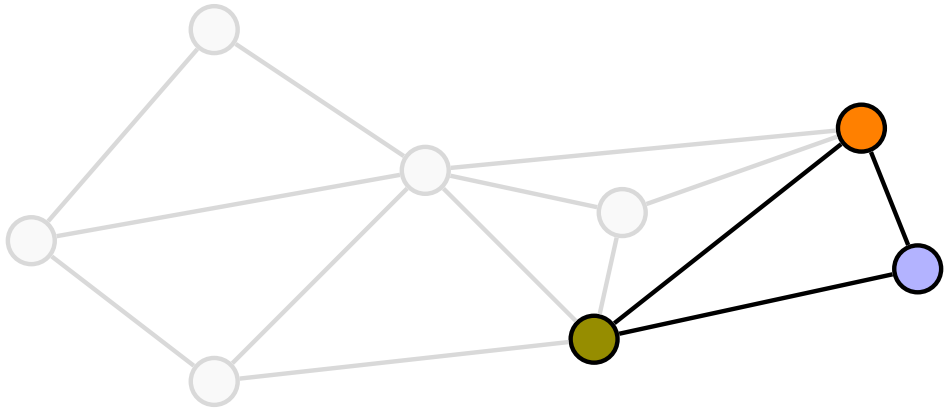
Forget

How to construct graphs



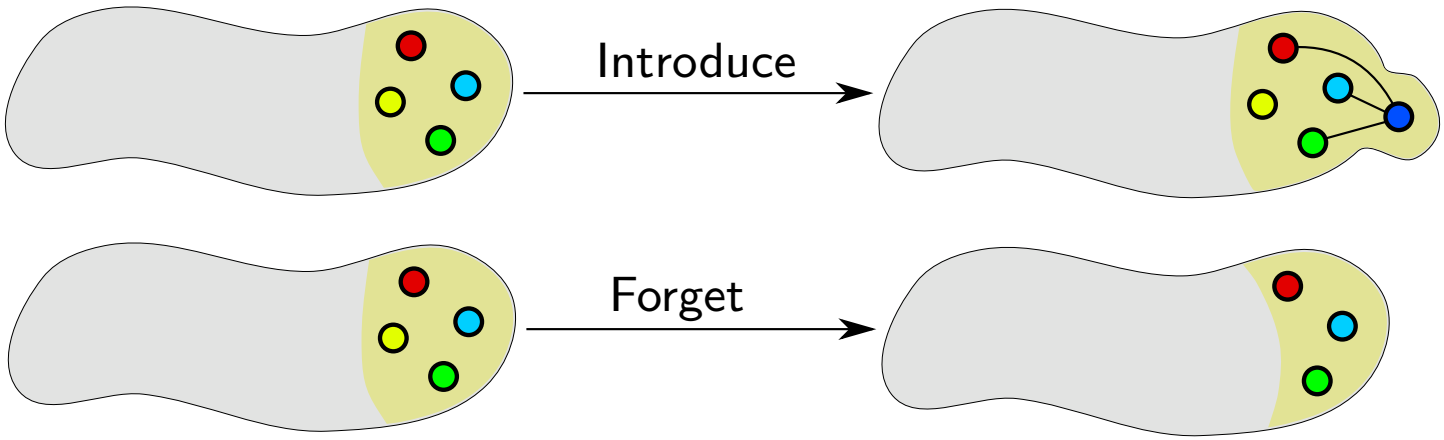
Forget

How to construct graphs

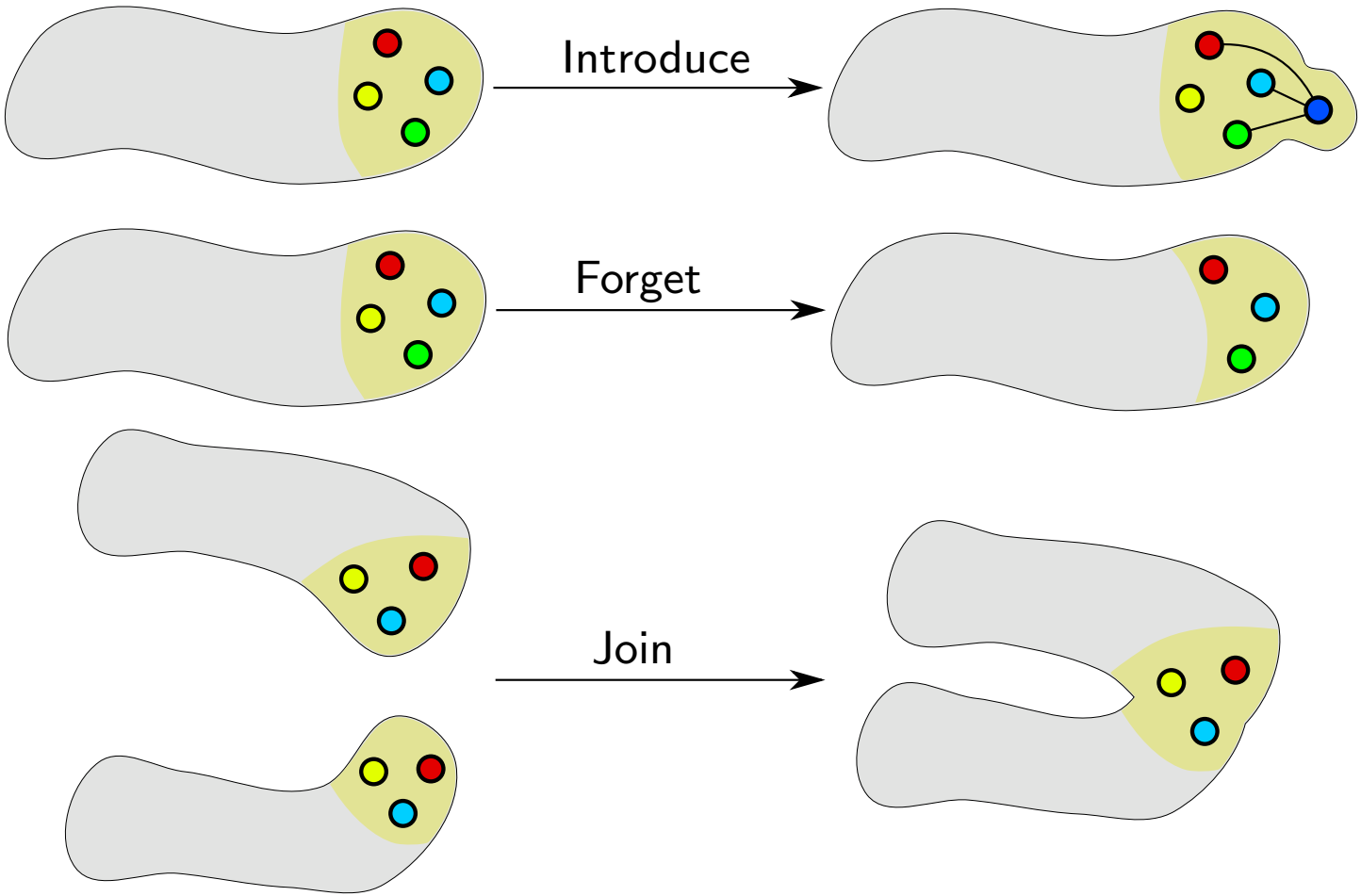


Introduce

Operations



Operations



Graph algebra

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

Graph algebra

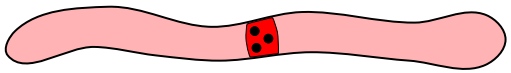
- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.

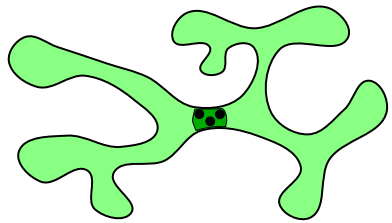
Graph algebra

- **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**.

Graph algebra

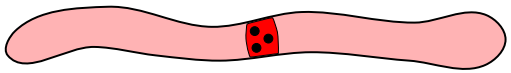
- **Algebra** $\mathbb{A}_k$: k-interface graphs with **introduce**, **forget**, and **join**.
- **Treewidth** of a graph G:
  the minimum k needed to construct G using all three operations.
- **Pathwidth** of a graph G:
  the minimum k needed to construct G using **introduce** and **forget**.

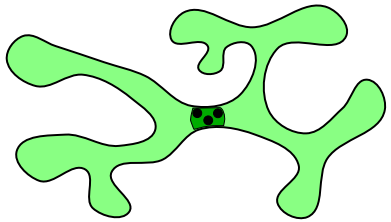




Graph algebra

- **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**.

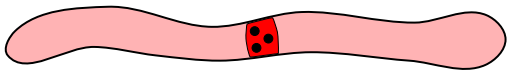


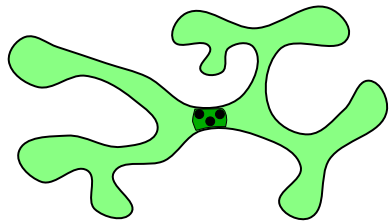


- Treewidth measures *tree-likeness*.
  Pathwidth measures *sausage-likeness*.

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**.

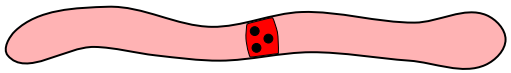


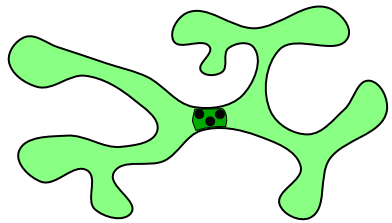


- Treewidth measures *tree-likeness*.
  Pathwidth measures *sausage-likeness*.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.

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**.

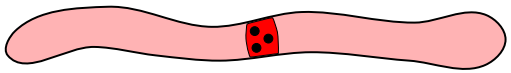


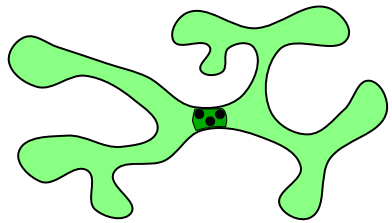


- Treewidth measures *tree-likeness*.
  Pathwidth measures *sausage-likeness*.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.
  - With each node associate its *bag*: the vertices active at the moment.

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**.



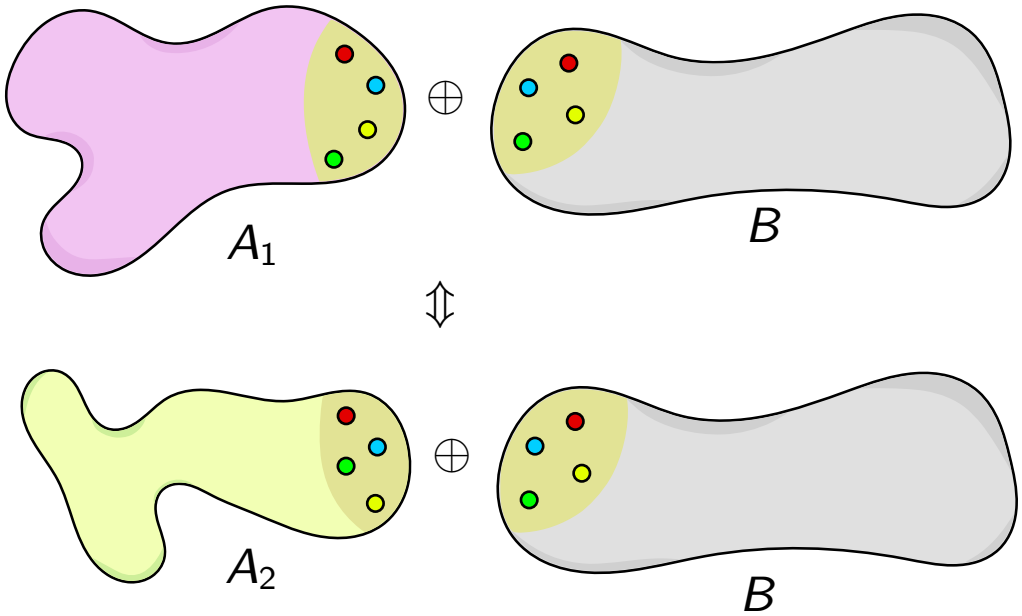


- Treewidth measures *tree-likeness*.
  Pathwidth measures *sausage-likeness*.
- **Tree decomposition**: the tree of the term over $\mathbb{A}_k$ constructing G.
  - With each node associate its *bag*: the vertices active at the moment.
  - The parameter k is the *width* of the decomposition.

Recognizability

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

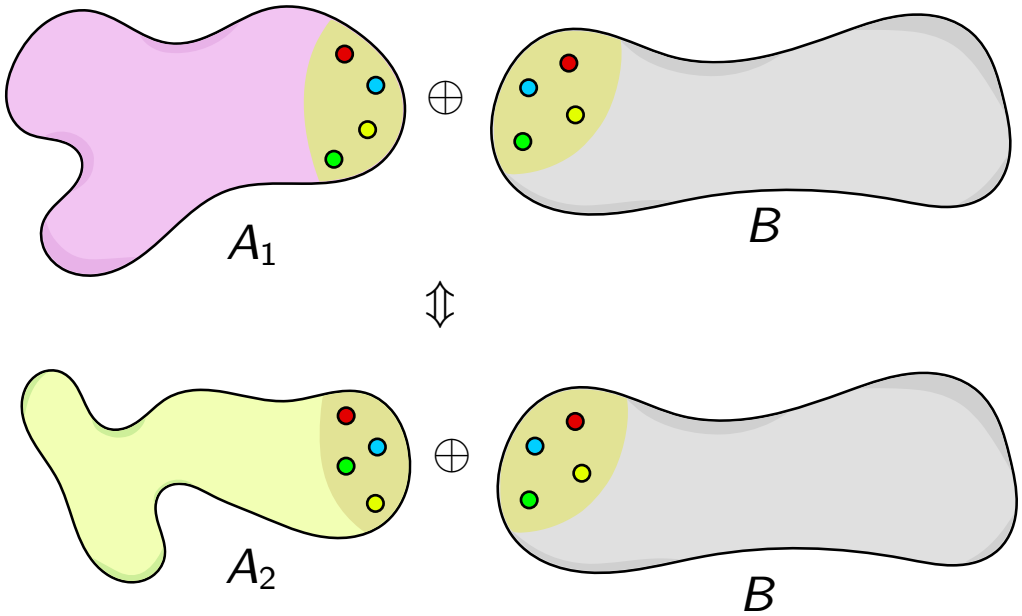
$$A_1 \equiv_k A_2 \quad \Leftrightarrow \quad A_1 \oplus B \in \Pi \text{ iff } A_2 \oplus B \in \Pi \text{ for every } k\text{-interface graph } B.$$



Recognizability

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

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

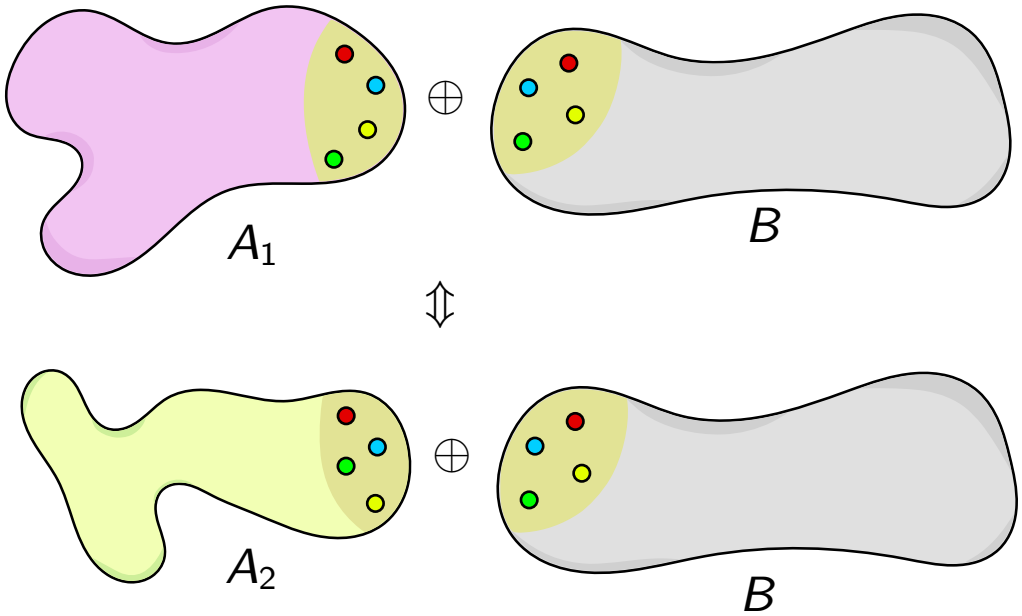


- Π is **recognizable** if it is k-recognizable for every k.

Recognizability

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

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



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

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.
  - Examples: 3-colorability, Hamiltonicity.

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.
  - Examples: 3-colorability, Hamiltonicity.
- **Question**: Does the converse hold?

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.
  - Examples: 3-colorability, Hamiltonicity.
- **Question**: Does the converse hold?

Courcelle's conjecture

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.

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.
  - Examples: 3-colorability, Hamiltonicity.
- **Question**: Does the converse hold?

Courcelle's conjecture

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.

- **Note**: For $k = 1$, this is essentially equivalence between MSO on trees and regular tree languages.

Definability

Theorem

Every MSO-definable property of graphs is recognizable.

- MSO on graphs: quantify over both vertex and edge subsets.
  - Examples: 3-colorability, Hamiltonicity.
- **Question**: Does the converse hold?

Courcelle's conjecture

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.

- **Note**: For $k = 1$, this is essentially equivalence between MSO on trees and regular tree languages.
- (BP,16^+): The conjecture indeed holds.

Attempt on the proof

- 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

- 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

- 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

- 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

- 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

- 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!

Theorem

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

Attempt on the proof

- 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!

Theorem

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

- **Thanks for attention!**