A finite presentation of graphs of treewidth at most 3

Amina Doumane, Humeau Samuel, Damien Pous

-

LIP, ENS de Lyon

12/07/2024

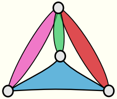
Outline

Two parts:

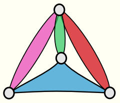
1. Basic notions and the main result: graphs, terms, and axioms.
2. Main tools of the proof, or how to decompose graphs.

Graphs, terms, treewidth, and the main result

Graphs

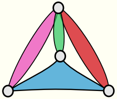


Graphs



Specificities:

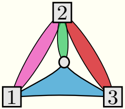
Graphs



Specificities:

- **Hyperedges**,

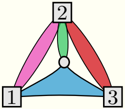
Graphs



Specificities:

- **Hyperedges**,

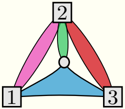
Graphs



Specificities:

- **Hyperedges**,
- **Numbered sources**: squared vertices, form the interface of the graph.

Graphs



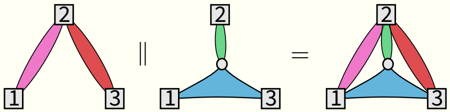
Specificities:

- **Hyperedges**,
- **Numbered sources**: squared vertices, form the interface of the graph.

Arity of a graph: number of sources.

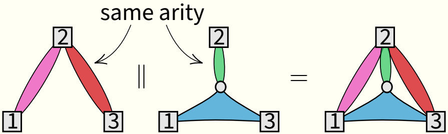
Graph operations

Parallel operation, $- \parallel -$:



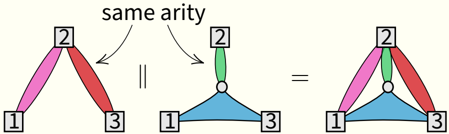
Graph operations

Parallel operation, $- \parallel -$:

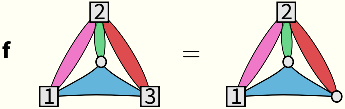


Graph operations

Parallel operation, $- \parallel -$:

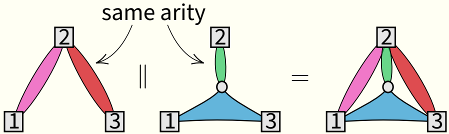


Forget operation, $\mathbf{f}(-)$:

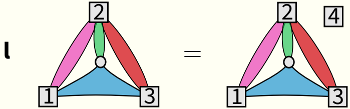


Graph operations

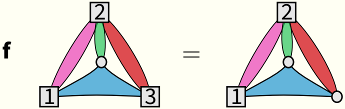
Parallel operation, $- \parallel -$:



Lift operation, $\mathbf{l}(-)$:

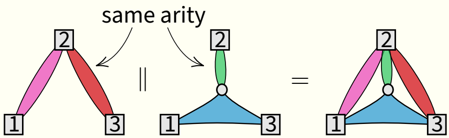


Forget operation, $\mathbf{f}(-)$:

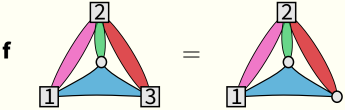


Graph operations

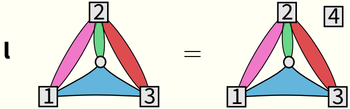
Parallel operation, $- \parallel -$:



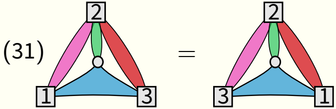
Forget operation, $\mathbf{f}(-)$:



Lift operation, $\mathbf{l}(-)$:

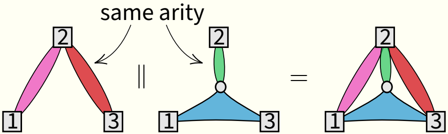


Permutation of sources, $p(-)$:

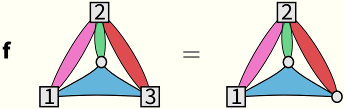


Graph operations

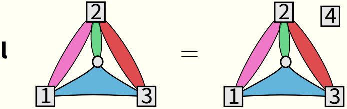
Parallel operation, $- \parallel -$:



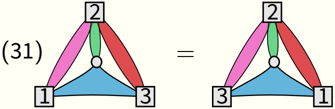
Forget operation, $\mathbf{f}(-)$:



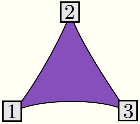
Lift operation, $\mathbf{l}(-)$:



Permutation of sources, $p(-)$:

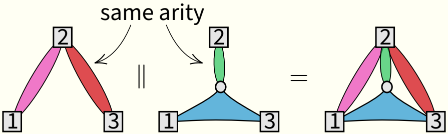


Edge of arity k, a:

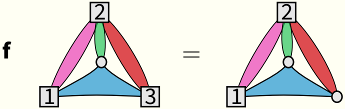


Graph operations

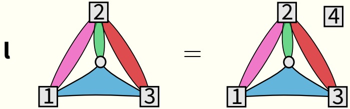
Parallel operation, $- \parallel -$:



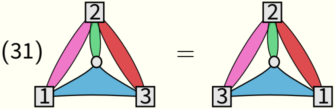
Forget operation, $\mathbf{f}(-)$:



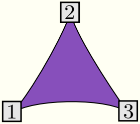
Lift operation, $\mathbf{l}(-)$:



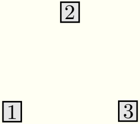
Permutation of sources, $p(-)$:



Edge of arity k, a:



Empty graph of arity k, $\emptyset$:



Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

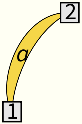
Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$



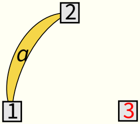
a

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$



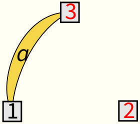
$\mathbf{l} a$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\,\mathbf{l}\, a)$



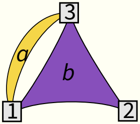
$(32)\,\mathbf{l}\, a$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\,\mathbf{l}\, a)$



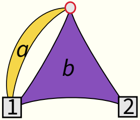
$b \parallel (32)\,\mathbf{l}\, a$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$



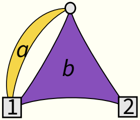
$\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\,\mathbf{l}\, a)$



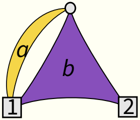
$\mathbf{f}(b \parallel (32)\,\mathbf{l}\, a)$

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$



$\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$

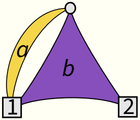
- Function giving the **graph of a term** noted $\mathbf{g}\, t$.

Terms

Corresponding grammar:

$$t, u ::= t \parallel u \mid \mathbf{l}\, t \mid \mathbf{f}\, t \mid \mathrm{p}\, t \mid a \mid \emptyset,$$

Example: $\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$

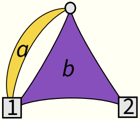


$\mathbf{f}(b \parallel (32)\, \mathbf{l}\, a)$

- Function giving the **graph of a term** noted $\mathbf{g}\, t$.
- **Parsing** of a graph G: term t such that $\mathbf{g}\, t \simeq G$.

Multiple parsings and goal

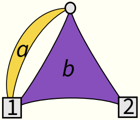
When do terms parse the same graph ?



$\mathbf{f}(b \parallel (32)\,\mathbf{l}\,a)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Multiple parsings and goal

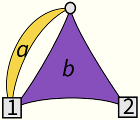
When do terms parse the same graph ?



$\mathbf{f}((32)(32)b \parallel (32)\,\mathbf{l}\,a)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Multiple parsings and goal

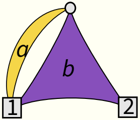
When do terms parse the same graph ?



$\mathbf{f}((32)(32)b \parallel (32)\,\mathbf{l}\,a)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Multiple parsings and goal

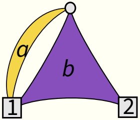
When do terms parse the same graph ?



$\mathbf{f}\,{\color{red}(32)}((32)b \parallel \mathbf{l}\,a)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Multiple parsings and goal

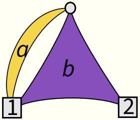
When do terms parse the same graph ?



$\mathbf{f}(32)((32)b \parallel \mathbf{l}\, a)$
$\mathbf{f}(32)(\mathbf{l}\, a \parallel (32)b)$

Multiple parsings and goal

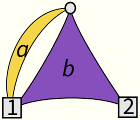
When do terms parse the same graph ?



$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Multiple parsings and goal

When do terms parse the same graph ?



$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Problem

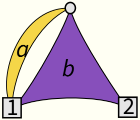
Is there a structured list of axioms Ax such that for all terms t, u

$$\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\,t \simeq \mathbf{g}\,u$$

holds ?

Multiple parsings and goal

When do terms parse the same graph ?



$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$
$\mathbf{f}(32)(\mathbf{l}\,a \parallel (32)b)$

Problem

Is there a structured list of axioms Ax such that for all terms t, u

$$\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\,t \simeq \mathbf{g}\,u$$

holds ?

Courcelle & Engelfriet, 2012:

- **infinite solution** for general graphs,
- **open question**: by restricting to graphs of bounded treewidth, can we find **finite** lists of associated axioms ?

Treewidth

Courcelle & Engelfriet, 2012:

- **infinite solution** for general graphs,
- **open question**: by restricting to graphs of bounded treewidth, can we find **finite** lists of associated axioms ?

Treewidth

Courcelle & Engelfriet, 2012:

- **infinite solution** for general graphs,
- **open question**: by restricting to graphs of bounded treewidth, can we find **finite** lists of associated axioms ?

Proposition

Graphs of terms t such that the maximum arity appearing in t is 4 are exactly graphs of treewidth at most 3.

Treewidth

Courcelle & Engelfriet, 2012:

- **infinite solution** for general graphs,
- **open question**: by restricting to graphs of bounded treewidth, can we find **finite** lists of associated axioms ?

Proposition

Graphs of terms t such that the maximum arity appearing in t is 4 are exactly graphs of treewidth at most 3.

To get bounded treewidth just bound the number of sources of terms

Treewidth

Courcelle & Engelfriet, 2012:

- **infinite solution** for general graphs,
- **open question**: by restricting to graphs of bounded treewidth, can we find **finite** lists of associated axioms ?

Proposition

Graphs of terms t such that the maximum arity appearing in t is 4 are exactly graphs of treewidth at most 3.

To get bounded treewidth just bound the number of sources of terms

We focus on graphs of treewidth at most 3.

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c$,

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a,$

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a$, and $\quad a \parallel \emptyset = a$,
2. $pqa = (p \circ q)a$, and $\quad \mathrm{id}\, a = a$,
3. $p(a \parallel b) = pa \parallel pb$ and $\quad p\emptyset = \emptyset$,
4. $\mathbf{l}(a \parallel b) = \mathbf{l}\, a \parallel \mathbf{l}\, b$ and $\quad \mathbf{l}\, \emptyset = \emptyset$,
5. $p\, \mathbf{f}\, a = \mathbf{f}\, \dot{p}a$ and $\quad \mathbf{l}\, pa = \dot{p}\, \mathbf{l}\, a$,
6. $\mathbf{l}\, \mathbf{f}\, a = \mathbf{f}\, r\, \mathbf{l}\, a$ and $\quad \mathbf{l}\, \mathbf{l}\, a = r\, \mathbf{l}\, \mathbf{l}\, a$,
7. $\mathbf{f}\, a \parallel b = \mathbf{f}(a \parallel \mathbf{l}\, b)$,
8. + 4 other axioms not detailed here.

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a$, and $\quad a \parallel \emptyset = a$,
2. $pqa = (p \circ q)a$, and $\quad \mathrm{id}\, a = a$,
3. $p(a \parallel b) = pa \parallel pb$ and $\quad p\emptyset = \emptyset$,
4. $\mathbf{l}(a \parallel b) = \mathbf{l}\, a \parallel \mathbf{l}\, b$ and $\quad \mathbf{l}\,\emptyset = \emptyset$,
5. $p\, \mathbf{f}\, a = \mathbf{f}\, \dot{p}a$ and $\quad \mathbf{l}\, pa = \dot{p}\, \mathbf{l}\, a$,
6. $\mathbf{l}\, \mathbf{f}\, a = \mathbf{f}\, r\, \mathbf{l}\, a$ and $\quad \mathbf{l}\, \mathbf{l}\, a = r\, \mathbf{l}\, \mathbf{l}\, a$,
7. $\mathbf{f}\, a \parallel b = \mathbf{f}(a \parallel \mathbf{l}\, b)$,
8. + 4 other axioms not detailed here.

Theorem

For all terms t, u of treewidth at most 3 $\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\, t \simeq \mathbf{g}\, u$.

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a$, and $\quad a \parallel \emptyset = a$,
2. $pqa = (p \circ q)a$, and $\quad \mathrm{id}\, a = a$,
3. $p(a \parallel b) = pa \parallel pb$ and $\quad p\emptyset = \emptyset$,
4. $\mathbf{l}(a \parallel b) = \mathbf{l}\, a \parallel \mathbf{l}\, b$ and $\quad \mathbf{l}\,\emptyset = \emptyset$,
5. $p\, \mathbf{f}\, a = \mathbf{f}\, \dot{p} a$ and $\quad \mathbf{l}\, pa = \dot{p}\, \mathbf{l}\, a$,
6. $\mathbf{l}\, \mathbf{f}\, a = \mathbf{f}\, r\, \mathbf{l}\, a$ and $\quad \mathbf{l}\, \mathbf{l}\, a = r\, \mathbf{l}\, \mathbf{l}\, a$,
7. $\mathbf{f}\, a \parallel b = \mathbf{f}(a \parallel \mathbf{l}\, b)$,
8. + 4 other axioms not detailed here.

Theorem

For all terms t, u of treewidth at most 3 $\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\, t \simeq \mathbf{g}\, u$.

Proof idea: we use axioms to recursively decompose terms following structural decompositions adapted to our graphs,

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a$, and $\quad a \parallel \emptyset = a$,
2. $pqa = (p \circ q)a$, and $\quad \mathrm{id}\, a = a$,
3. $p(a \parallel b) = pa \parallel pb$ and $\quad p\emptyset = \emptyset$,
4. $\mathbf{l}(a \parallel b) = \mathbf{l}\, a \parallel \mathbf{l}\, b$ and $\quad \mathbf{l}\, \emptyset = \emptyset$,
5. $p\, \mathbf{f}\, a = \mathbf{f}\, \dot{p}a$ and $\quad \mathbf{l}\, pa = \dot{p}\, \mathbf{l}\, a$,
6. $\mathbf{l}\, \mathbf{f}\, a = \mathbf{f}\, r\, \mathbf{l}\, a$ and $\quad \mathbf{l}\, \mathbf{l}\, a = r\, \mathbf{l}\, \mathbf{l}\, a$,
7. $\mathbf{f}\, a \parallel b = \mathbf{f}(a \parallel \mathbf{l}\, b)$,
8. + 4 other axioms not detailed here.

Theorem

For all terms t, u of treewidth at most 3 $\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\, t \simeq \mathbf{g}\, u$.

Proof idea: we use axioms to recursively decompose terms following structural decompositions adapted to our graphs, so two sides:

Structural decompositions of graphs $\quad$ Apply them on terms using axioms

Main result and associated method

Axioms Ax given by:

1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c, \quad a \parallel b = b \parallel a$, and $\quad a \parallel \emptyset = a$,
2. $pqa = (p \circ q)a$, and $\quad \mathrm{id}\, a = a$,
3. $p(a \parallel b) = pa \parallel pb$ and $\quad p\emptyset = \emptyset$,
4. $\mathbf{l}(a \parallel b) = \mathbf{l}\, a \parallel \mathbf{l}\, b$ and $\quad \mathbf{l}\,\emptyset = \emptyset$,
5. $p\,\mathbf{f}\,a = \mathbf{f}\,\dot{p}a$ and $\quad \mathbf{l}\,pa = \dot{p}\,\mathbf{l}\,a$,
6. $\mathbf{l}\,\mathbf{f}\,a = \mathbf{f}\,r\,\mathbf{l}\,a$ and $\quad \mathbf{l}\,\mathbf{l}\,a = r\,\mathbf{l}\,\mathbf{l}\,a$,
7. $\mathbf{f}\,a \parallel b = \mathbf{f}(a \parallel \mathbf{l}\,b)$,
8. + 4 other axioms not detailed here.

Theorem

For all terms t, u of treewidth at most 3 $\mathrm{Ax} \vdash t = u \Leftrightarrow \mathbf{g}\,t \simeq \mathbf{g}\,u$.

Proof idea: we use axioms to recursively decompose terms following structural decompositions adapted to our graphs, so two sides:

Structural decompositions of graphs ~~Apply them on terms using axioms~~

Recursively decomposing graphs of treewidth at most 3

A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators

A usual decomposition in graph theory

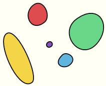
Reduce to more and more connected graphs by identifying separators:

- A **graph** is the disjoint union of its connected components

A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators:

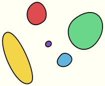
- A **graph** is the disjoint union of its connected components:



A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators:

- A **graph** is the disjoint union of its connected components:

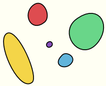


- A connected graph has a tree structure when looking only at 1-separators, or cutvertices

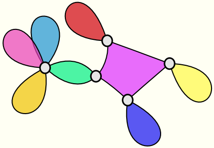
A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators:

- A **graph** is the disjoint union of its connected components:



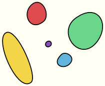
- A connected graph has a tree structure when looking only at 1-separators, or cutvertices:



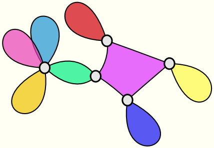
A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators:

- A **graph** is the disjoint union of its connected components:



- A connected graph has a tree structure when looking only at 1-separators, or cutvertices:

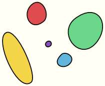


- A biconnected graph still admits a canonical decomposition along its 2-separators (*Tutte*, 1961).

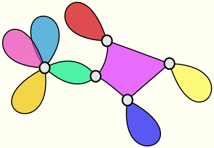
A usual decomposition in graph theory

Reduce to more and more connected graphs by identifying separators:

- A **graph** is the disjoint union of its connected components:



- A connected graph has a tree structure when looking only at 1-separators, or cutvertices:

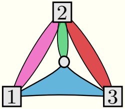


- A biconnected graph still admits a canonical decomposition along its 2-separators (*Tutte*, 1961).

We adapt these decompositions to sourced graphs.

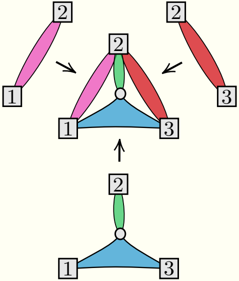
Decomposing graphs: full prime decompositions

Consider the following graph:



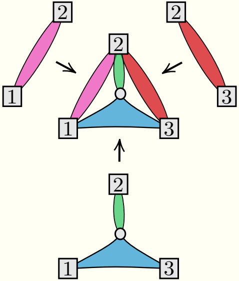
Decomposing graphs: full prime decompositions

Consider the following graph:



Decomposing graphs: full prime decompositions

Consider the following graph:

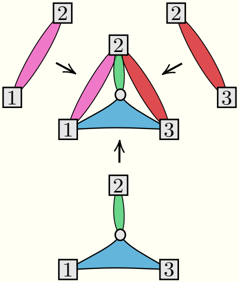


Basic simple graphs

- **Paths** between vertices,

Decomposing graphs: full prime decompositions

Consider the following graph:

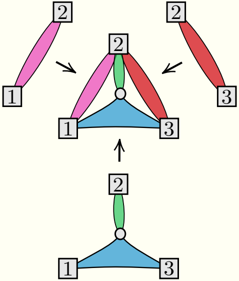


Basic simple graphs

- **Paths** between vertices,
- **Connected components**,

Decomposing graphs: full prime decompositions

Consider the following graph:

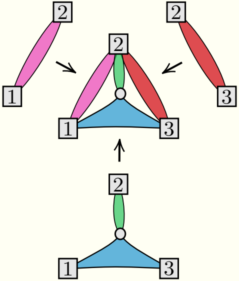


Basic simple graphs

- **Paths** between vertices,
- **Connected components**,
- **Proposition:** for a graph G, $G \simeq \uplus_i H_i$ with H_i its connected components.

Decomposing graphs: full prime decompositions

Consider the following graph:



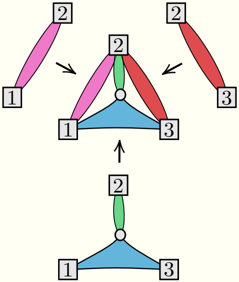
Basic simple graphs

- **Paths** between vertices,
- **Connected components**,
- **Proposition:** for a graph G, $G \simeq \uplus_i H_i$ with H_i its connected components.

Sourced graphs

Decomposing graphs: full prime decompositions

Consider the following graph:



Basic simple graphs

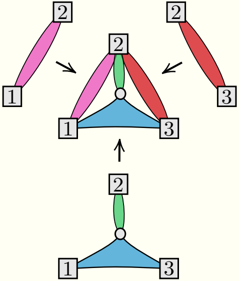
- **Paths** between vertices,
- **Connected components**,
- **Proposition:** for a graph G, $G \simeq \uplus_i H_i$ with H_i its connected components.

Sourced graphs

- **Paths without sources**, except at endpoints,

Decomposing graphs: full prime decompositions

Consider the following graph:



Basic simple graphs

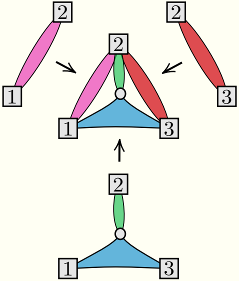
- **Paths** between vertices,
- **Connected components**,
- **Proposition:** for a graph G, $G \simeq \uplus_i H_i$ with H_i its connected components.

Sourced graphs

- **Paths without sources**, except at endpoints,
- **Full prime** components,

Decomposing graphs: full prime decompositions

Consider the following graph:



Basic simple graphs

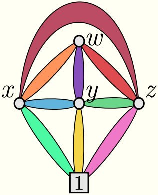
- **Paths** between vertices,
- **Connected components**,
- **Proposition:** for a graph G, $G \simeq \uplus_i H_i$ with H_i its connected components.

Sourced graphs

- **Paths without sources**, except at endpoints,
- **Full prime** components,
- **Proposition:** for a graph G, $G \simeq \|_i \; p_i \, \mathbf{l}^{j_i} \, H_i$ with H_i its full prime components.

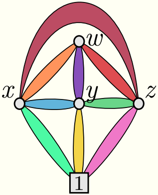
The main difficulty: forget points non-determinism

A full prime graph of treewidth 3:



The main difficulty: forget points non-determinism

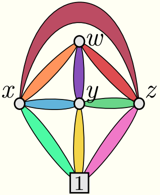
A full prime graph of treewidth 3:



- making w a source gives a graph of treewidth 4 (because we would have K_5),

The main difficulty: forget points non-determinism

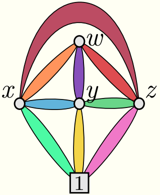
A full prime graph of treewidth 3:



- making w a source gives a graph of treewidth 4 (because we would have K_5),
- making x a source gives a graph of treewidth 3,

The main difficulty: forget points non-determinism

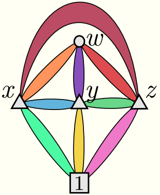
A full prime graph of treewidth 3:



- making w a source gives a graph of treewidth 4 (because we would have K_5),
- making x a source gives a graph of treewidth 3,
- and similarly for y and z.

The main difficulty: forget points non-determinism

A full prime graph of treewidth 3:

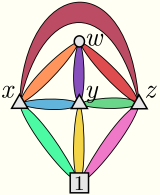


- making w a source gives a graph of treewidth 4 (because we would have K_5),
- making x a source gives a graph of treewidth 3,
- and similarly for y and z.

x, y, and z (but not w) are called **forget points** (shown as triangles).

The main difficulty: forget points non-determinism

A full prime graph of treewidth 3:



- making w a source gives a graph of treewidth 4 (because we would have K_5),
- making x a source gives a graph of treewidth 3,
- and similarly for y and z.

x, y, and z (but not w) are called **forget points** (shown as triangles).

The difficulty comes from graphs having many forget points: there are no canonical choices.

Decomposing graphs: anchors

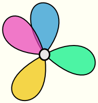
Decomposing graphs: anchors

Basic simple graphs:
Cutvertices: vertices such that removing them disconnects the graph:

Decomposing graphs: anchors

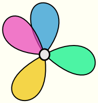
Basic simple graphs:

Cutvertices: vertices such that removing them disconnects the graph: (petals are parts of the graph)



Decomposing graphs: anchors

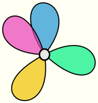
Basic simple graphs:
Cutvertices: vertices such that removing them disconnects the graph: (petals are parts of the graph)



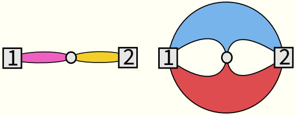
Sourced graphs:
Anchors:

Decomposing graphs: anchors

Basic simple graphs:
Cutvertices: vertices such that removing them disconnects the graph: (petals are parts of the graph)

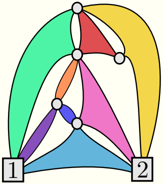


Sourced graphs:
Anchors: (here for arity 2)



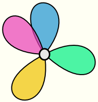
+1 case

Decomposing graphs: anchors



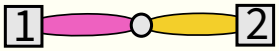
Basic simple graphs:

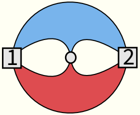
Cutvertices: vertices such that removing them disconnects the graph: (petals are parts of the graph)



Sourced graphs:

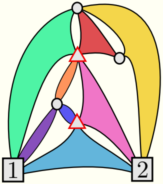
Anchors: (here for arity 2)



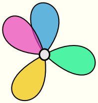


$+1$ case

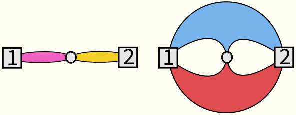
Decomposing graphs: anchors



Basic simple graphs:
Cutvertices: vertices such that removing them disconnects the graph: (petals are parts of the graph)



Sourced graphs:
Anchors: (here for arity 2)



+1 case

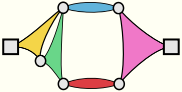
Decomposing graphs: separation pairs

Hard graph: a full prime graph without anchors.

Decomposing graphs: separation pairs

Hard graph: a full prime graph without anchors.

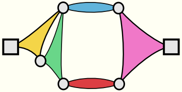
An example:



Decomposing graphs: separation pairs

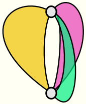
Hard graph: a full prime graph without anchors.

An example:



Basic simple graphs:

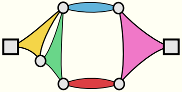
2-separators: pairs of vertices disconnecting the graph:



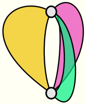
Decomposing graphs: separation pairs

Hard graph: a full prime graph without anchors.

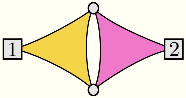
An example:



Basic simple graphs:
2-separators: pairs of vertices disconnecting the graph:



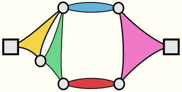
Sourced graphs:
Separation pair:



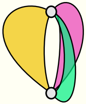
Decomposing graphs: separation pairs

Hard graph: a full prime graph without anchors.

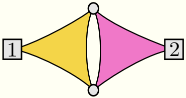
An example:



Basic simple graphs:
2-separators: pairs of vertices disconnecting the graph:



Sourced graphs:
Separation pair:



Proposition

Every hard graph of treewidth at most 3 has at least one (minimal) separation pair consisting of forget points.

Classifying hard graphs of treewidth at most 3

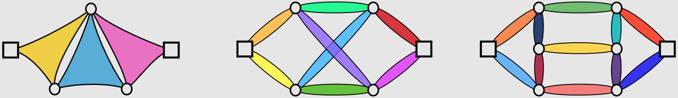
What happens if a hard graph has several separation pairs ?

Classifying hard graphs of treewidth at most 3

What happens if a hard graph has several separation pairs ?

Theorem

Whenever a hard graph has two distinct (minimal) separation pairs whose vertices are forget points, then it must have one of the following shapes:



An important lemma

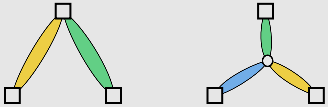
The proof of the last theorem requires a case analysis handling lots of possibilities.

An important lemma

The proof of the last theorem requires a case analysis handling lots of possibilities. The following lemma reduces their number:

Lemma

A graph with 3 sources either has the triangle K_3 as a minor on its sources, or it has one of the following shapes:

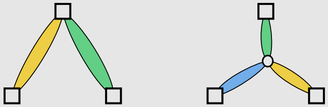


An important lemma

The proof of the last theorem requires a case analysis handling lots of possibilities. The following lemma reduces their number:

Lemma

A graph with 3 sources either has the triangle K_3 as a minor on its sources, or it has one of the following shapes:



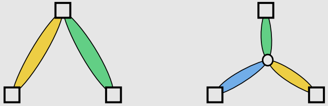
Graph minors: a way to see patterns in graphs, generalising the notion of subgraphs.

An important lemma

The proof of the last theorem requires a case analysis handling lots of possibilities. The following lemma reduces their number:

Lemma

A graph with 3 sources either has the triangle K_3 as a minor on its sources, or it has one of the following shapes:



Graph minors: a way to see patterns in graphs, generalising the notion of subgraphs.
This is reminiscent of the proposition stating that a graph without a cycle (i.e. without K_3 as a minor) is a tree. Our lemma is stronger.

Conclusion

Generalising ?

Generalising to treewidth at most 4 ?

Generalising ?

Generalising to treewidth at most 4 ?

- we can go up to 3-separators but no more,

Generalising ?

Generalising to treewidth at most 4 ?

- we can go up to 3-separators but no more,
- the number of analysed cases of the problem explodes,

Generalising ?

Generalising to treewidth at most 4 ?

- we can go up to 3-separators but no more,
- the number of analysed cases of the problem explodes,
- the main tool to reduce it here is the last lemma, which would need to be generalised for K_4 instead of K_3,

Generalising ?

Generalising to treewidth at most 4 ?

- we can go up to 3-separators but no more,
- the number of analysed cases of the problem explodes,
- the main tool to reduce it here is the last lemma, which would need to be generalised for K_4 instead of K_3,
- but such a generalisation is not possible.

Algorithmic applications ?

Two possible **algorithmic applications**:

Algorithmic applications ?

Two possible **algorithmic applications**:

- testing treewidth at most 3,
- solving the isomorphism problem for graphs of treewidth at most 3

Algorithmic applications ?

Two possible **algorithmic applications**:

- testing treewidth at most 3,
- solving the isomorphism problem for graphs of treewidth at most 3.
- no polynomial algorithms are known for these problems when treewidth is not fixed

Algorithmic applications ?

Two possible **algorithmic applications**:

- testing treewidth at most 3,
- solving the isomorphism problem for graphs of treewidth at most 3.
- no polynomial algorithms are known for these problems when treewidth is not fixed,
- linear algorithms are already known for both problems and treewidth at most 3 graphs

Algorithmic applications ?

Two possible **algorithmic applications**:

- testing treewidth at most 3,
- solving the isomorphism problem for graphs of treewidth at most 3.
- no polynomial algorithms are known for these problems when treewidth is not fixed,
- linear algorithms are already known for both problems and treewidth at most 3 graphs,

The classification result might be usefull to get heuristic to compute treewidth in general.

Summary

Reminders:

- graphs with sources and hyperedges,
- syntax for bounded treewidth,
- axioms for treewidth at most 3,
- proven by structural analysis (full prime decompositions, anchors, and separation pairs),
- two research lines: generalisation and algorithmic aspects (testing treewidth at most 3 and the isomorphism on graphs of treewidth at most 3).

Summary

Reminders:

- graphs with sources and hyperedges,
- syntax for bounded treewidth,
- axioms for treewidth at most 3,
- proven by structural analysis (full prime decompositions, anchors, and separation pairs),
- two research lines: generalisation and algorithmic aspects (testing treewidth at most 3 and the isomorphism on graphs of treewidth at most 3).

Thank you !