Definability equals recognizability for graphs of bounded treewidth *

Mikołaj Bojańczyk Michał Pilipczuk
Institute of Informatics, University of Warsaw
{bojan,michal.pilipczuk}@mimuw.edu.pl

Abstract

We prove a conjecture of Courcelle, which states that a graph property is definable in MSO with modular counting predicates on graphs of constant treewidth if, and only if it is recognizable in the following sense: constant-width tree decompositions of graphs satisfying the property can be recognized by tree automata. While the forward implication is a classic fact known as Courcelle's theorem, the converse direction remained open.

Categories and Subject Descriptors F.4 [*Theory of Computation*]: Mathematical Logic and Formal Languages

Keywords treewidth, tree decomposition, Monadic Second-Order Logic, recognizability, Simon's factorization forest

1. Introduction

Classical results of Büchi, Trakhtenbrot and Elgot say that for finite words, languages recognised by finite automata are exactly those definable in Monadic Second-Order logic MSO. Courcelle's theorem shows the right-to-left inclusion holds for graphs of bounded treewidth: if a property of graphs can be defined in MSO with quantification over edge subsets and modular counting predicates — henceforth called counting MSO on graphs — then for every k there is a tree automaton recognising tree decompositions of width k of graphs satisfying the property. A corollary is that model checking counting MSO on graphs of constant treewidth can be done in linear time, which is one of the foundational results in parameterized complexity. For more details, we refer the reader to the monograph of Courcelle and Engelfriet [7] devoted to MSO on graphs.

Courcelle's theorem, stated as above, generalizes only one direction of the equivalence between MSO and automata. The converse question is whether one can use counting MSO to define any property of graphs that is recognizable for constant treewidth, where recognizability is defined by, say, the finiteness of the index of an appropriate Myhill-Nerode relation. In the case of words (or trees), this is the 'easy' direction: a formula of MSO can guess an accepting run of an automaton, by labelling nodes with states. Surprisingly, the generalization of this implication to graphs of constant treewidth remained open for the last 25 years.

This problem, known as the *Courcelle's conjecture*, was initially formulated by Courcelle in the very first paper of his monumental series *The monadic second-order logic of graphs* [4]. The fifth paper of the series [5] is entirely devoted to its investigation and contains a proof for graphs of treewidth 2. Since then, the conjecture has been confirmed for graphs of treewidth 3 [12], for k-connected graphs of treewidth k [12], for graphs of constant treewidth and chordality [1], and for k-outerplanar graphs [10]. There were also two claims of a significant progress on the general case. First, Kabanets [11] claimed a proof for graphs of bounded pathwidth, and then a resolution of the full conjecture was claimed by Lapoire [14]. Unfortunately, both these works are published only as extended conference abstracts, and no verified journal version has appeared. The proofs of Kabanets and Lapoire are widely regarded as unsatisfactory; cf. [1, 7, 8, 10]. In particular, the problem is stated as open both in the monograph of Courcelle and Engelfriet [7] and of Downey and Fellows [8].

The issue at the heart of Courcelle's conjecture is that an MSO formula is applied to the graph alone, without access to any predefined tree decomposition. Hence, one cannot simply guess a run of a tree automaton, because there is no tree. As Courcelle puts it in [5], *It is not clear at all how an automaton should traverse a graph. A "general" graph has no evident structure, whereas a word or a tree is (roughly speaking) its own algebraic structure*. Hence, the natural approach to proving the conjecture is to find, using MSO and the graph structure only, some tree decomposition of bounded width. This strategy, proposed by Courcelle [5], was used in all the previous work on Courcelle's conjecture. We also use this strategy.

Our main result is that there exists an MSO *transduction* that, given a graph of treewidth k encoded as a relational structure, outputs an encoding of a tree decomposition of width $f(k)$, for some function f. Informally speaking, an MSO transduction guesses existentially a number of vertex and edge subsets, and based on them defines a tree decomposition. Different guesses may lead to different decompositions, but provided the input graph has treewidth at most k, the output for at least one guess will be a tree decomposition of width bounded by $f(k)$. The precise statement is in Section 2.2.

Acknowledgment. We would like to thank Christoph Dittmann and Stephan Kreutzer for many helpful discussions.

2. Overview

The main technical result of this paper, stated in Theorem 2.4, is that using MSO one can compute a tree decomposition of a graph. This section describes the proof plan. We explain what it means to compute something in MSO, and divide Theorem 2.4 into lemmas.

* M. Bojańczyk is supported by the ERC Consolidator Grant LIPA. This work was partially done while Mi. Pilipczuk held a post-doc position at Warsaw Centre of Mathematics and Computer Science. Mi. Pilipczuk is also supported by the Foundation for Polish Science (FNP) via the START stipend programme.

LICS '16, July 05-08, 2016, New York, NY, USA

ACM 978-1-4503-4391-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2933575.2934508

2.1 Tree decompositions

In this section we define tree decompositions and show how we represent them for the purposes of MSO. Similar formalisms were used in the previous works, cf. [5, 10, 12], but we choose to introduce our own language for the sake of being self-contained.

Logical terminology. Define a *vocabulary* to be a set of relation names with associated arities (we do not use functions or constants). A *logical structure* over a vocabulary consists of a universe supplied with interpretations of relations in the vocabulary. We use logical structures to model things like graphs and tree decompositions.

Graphs as logical structures. We model (undirected) graphs as logical structures, where the universe consists of both vertices and edges, and there is a binary *incidence* relation $\mathsf{incident}(v, e)$ which says when a vertex v is incident with an edge e. The edges can be recovered as those elements of the universe that are second arguments of the incidence relation, and the vertices can be recovered as those elements of the universe that are not edges. We do not allow multiple edges connecting the same pair of vertices, i.e., all graphs considered in this work are simple, unless explicitly stated. We choose this encoding so that set quantification in MSO can capture sets of edges as well.

Tree decompositions. We begin by defining tree decompositions. Define an *in-forest* to be an acyclic directed graph where every node has outdegree at most one. We use the usual tree terminology: root, parent, and child. Every connected component of an in-forest is a tree with exactly one root (vertex of outdegree zero).

Definition 2.1. A *tree decomposition* of a graph G is an in-forest t whose vertices called *nodes*, and a labelling of the nodes by sets of vertices in G, called *bags*, subject to the following conditions:

- for every edge e of G, some bag contains both endpoints of e;
- for every vertex v of G, the set of nodes in t that are labelled by bags containing v is nonempty and connected in t.

Note two minor changes with respect to the classic definition: tree decompositions are rooted, and we allow them to be forests, instead of just trees. Both changes are for convenience only and bear no significance for our results. We write x, y, z for nodes and u, v, w for vertices. Below we introduce some terminology for tree decompositions, inspired by Grohe and Marx [9].

Definition 2.2. Let x be a node in a tree decomposition t. The *adhesion of* x is the intersection of the bags of x and its parent; if x is a root the adhesion is empty. The *margin of* x is its bag minus its adhesion. The *cone of* x is the union of the bags of the descendants of x, including x. The *component of* x is its cone minus its adhesion. If the decomposition t is not clear from the context, we write t-cone, t-adhesion, etc.

Note that the margins of the nodes of a tree decomposition form a partition of the vertex set of the underlying graph.

A *path decomposition* is the special case when the forest is a set of paths. The *width* of a tree or path decomposition is the maximum size of its bags, minus 1. The *treewidth* of a graph is the minimum possible width of its tree decomposition, likewise for *pathwidth*. The treewidth and pathwidth of a graph G are denoted by $\mathtt{tw}(G)$ and $\mathtt{pw}(G)$, respectively.

Tree decompositions as logical structures. A tree decomposition is represented as a logical structure as follows. The universe of the logical structure consists of the vertices and edges of the underlying graph, plus the nodes of the tree decomposition. The vocabulary, which we call the *vocabulary of tree decompositions*, consists of:

$$\underbrace{\mathsf{node}}_{\text{a unary predicate}} \qquad \underbrace{\mathsf{incident} \quad \mathsf{bag} \quad \mathsf{parent}}_{\text{binary predicates}}$$

The predicate incident describes the incidence relation in the underlying graph. The predicate $\mathsf{node}(x)$ says that x is a decomposition node, $\mathsf{bag}(v, x)$ says that vertex v is in the bag of node x, and $\mathsf{parent}(x, y)$ says that node x is the parent of node y.

MSO ***interpretations.*** We now define what it means to produce a tree decomposition (or some other structure) using MSO. We do this by using three types of basic operations on logical structures defined below, called *copying*, *coloring*, and *interpreting*. All three types describe binary relations on logical structures: copying is a function, coloring is a relation, and interpreting is a partial function.

1. *Copying.* Define the *k-copy* of a logical structure $\mathfrak{A}$ to be k disjoint copies of $\mathfrak{A}$, with the following fresh predicates added to the vocabulary:

   $$\mathsf{copy}(a, b)\ ,\ \mathsf{layer}_1(a)\ ,\ \ldots\ ,\ \mathsf{layer}_k(a).$$

   The binary predicate copy checks whether two elements are copies of the same element of the original structure, whereas the unary predicate layer_i checks whether an element belongs to the i-th copy (called also the i-th *layer*).

2. *Coloring.* Define an *k-coloring* of a structure $\mathfrak{A}$ to be any structure obtained from $\mathfrak{A}$ by adding new unary predicates $X_1, \ldots, X_k$ to the vocabulary and interpreting them as any subsets of the universe.

3. *Interpreting.* The syntax of an interpretation consists of an *input vocabulary* Σ, an *output vocabulary* Γ and a family of MSO formulas

   $$\{\varphi_{\mathsf{dom}}, \varphi_{\mathsf{univ}}\} \cup \{\varphi_R\}_{R \in \Gamma},$$

   over the input vocabulary Σ. The formula φ_{dom} has no free variables, the formula φ_{univ} has one free variable, and each formula φ_R has as many free variables as the arity of R. The free variables in all of these formulas range over elements, not sets of elements. If $\mathfrak{A}$ is a logical structure over the input vocabulary Σ that satisfies φ_{dom}, we define the output logical structure, which is over the output vocabulary Γ, as follows. The universe is the universe of $\mathfrak{A}$ restricted to elements satisfying φ_{univ} and each relation R of the output vocabulary is interpreted as those tuples in the universe which make φ_R true. If φ_{dom} is not satisfied in $\mathfrak{A}$, then the output of the interpretation is undefined.

Definition 2.3. An MSO *transduction* is a finite composition of the three types of operation defined above (treated as relations between logical structures), together with prescribed input and output vocabularies. If coloring is not used, we talk about a *deterministic* MSO transduction. If $\mathcal{I}$ is an MSO transduction, and $\mathfrak{A}$ is a structure over the input vocabulary, then by $\mathcal{I}(\mathfrak{A})$ we denote the *output* of $\mathcal{I}$, defined as the set of all structures over the output vocabulary that are in relation defined by $\mathcal{I}$ with $\mathfrak{A}$.

The definition above is equivalent to the one in [6]; this equivalence follows from [6, Theorem 1.39]. The crucial property of MSO transductions is the Backwards Translation Theorem [6, Theorem 1.40], which says that if $\mathcal{I}$ a MSO transduction and ψ is an MSO sentence over the output vocabulary, then

$$\{\mathfrak{A} : \text{at least on structure in } \mathcal{I}(\mathfrak{A}) \text{ satisfies } \psi\}$$

is a set of structures over the input vocabulary that is definable in MSO. Using this result, we may apply MSO transductions to enrich the input structure with MSO-definable objects, and any property that can be defined in MSO afterwards, can be also defined directly in the input structure.

2.2 The main result

We are now ready to state our main technical result, which says that an MSO transduction can compute tree decompositions for graphs

of bounded treewidth. We use the name *transduction from graphs to tree decompositions* if the input vocabulary is the vocabulary of graphs $\{\mathsf{incident}(x, y)\}$ and the output vocabulary is the vocabulary of tree decompositions defined previously.

Theorem 2.4. *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that for every $k \in \mathbb{N}$ there exists an* MSO *transduction $\mathcal{I}$ from graphs to tree decompositions, such that every graph G satisfies:*

(1) Every output $\mathcal{I}(G)$ represents a tree decomposition of G of width at most $f(k)$.
(2) If G has treewidth at most k, then the output $\mathcal{I}(G)$ is nonempty.

We actually believe that a stronger variant of the above theorem holds, with f being the identity. In other words, we believe that there is an MSO transduction which inputs a graph of treewidth k, and produces a tree decomposition of width k. In order to prove the stronger version, it would be sufficient to show that for every $k \leq k'$, there is an MSO transduction which realizes the following task: given a tree decomposition of width k', produce, if possible, a tree decomposition of width k for the same graph. Our idea for a proof of this statement is to take a closer look at the algorithm of Bodlaender and Kloks [2] that solves exactly this task, and try to simulate it using a deterministic MSO transduction. We expand this topic in the concluding section of the full version of the paper.

The proof of Theorem 2.4 consists of two steps, described below.

The special case of bounded pathwidth. The first step is to prove a weaker variant of the theorem. This variant has exactly the same statement, except that in condition (2) the assumptions are strengthened to requiring that the pathwidth of the graph is at most k. This weaker variant, Lemma 2.5 below, is proved in Section 4.

Lemma 2.5. *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that for every $k \in \mathbb{N}$ there exists an* MSO *transduction $\mathcal{I}$ from graphs to tree decompositions, such that every graph G satisfies:*

(1) Every output $\mathcal{I}(G)$ represents a tree decomposition of G of width at most $f(k)$.
(2) If G has pathwidth at most k, then the output $\mathcal{I}(G)$ is nonempty.

There are two crucial ingredients in the proof of Lemma 2.5.

The first ingredient is that for path decompositions, we can use semigroup theory. Specifically, we use Factorisation Forest Theorem of Imre Simon [15]. The application of this result is the cornerstone of our approach, and it enables us to recursively decompose any graph of pathwidth k into constant-size pieces using only $f(k)$ levels of recursion — a number that depends on k alone, and not on the size of the graph. Lemma 2.5 then follows by verifying that each level incurs a fixed blow-up of the width of tree decompositions that we are able to describe in MSO.

The second ingredient is the definition of a *guidance system*, which is a combinatorial object used to describe additional structure in a graph, e.g., a tree decomposition, in a way that can be guessed by an MSO transduction. Guidance systems are introduced in Section 3, and are used throughout the whole paper to describe "MSO-guessable" tree decompositions.

Tree decompositions with bags of bounded pathwidth. In the second step, presented in Section 5, we show that there is an MSO transduction which inputs a graph of treewidth at most k, and outputs a tree decomposition of the graph where the bags are maybe arbitrarily large, but have pathwidth bounded by $2k + 1$, in the following sense. For a node x in a tree decomposition t, define its *marginal graph* to be subgraph of the underlying graph induced by the margin of x, with edge uv added for every pair of vertices $\{u, v\}$ that appear together in the adhesion of some child node of x.

Lemma 2.6. *For every $k \in \mathbb{N}$, there exists an* MSO *transduction $\mathcal{B}$ from graphs to tree decompositions such that for every graph G the following holds:*

(1) Every output $\mathcal{B}(G)$ represents a sane tree decomposition of G.
(2) If G has treewidth at most k, then $\mathcal{B}(G)$ contains at least one tree decomposition where all marginal graphs have pathwidth at most $2k + 1$.

In Lemma 2.6 we use a technical notion of a *sane tree decomposition*, which is defined below.

Definition 2.7. A tree decomposition t of a graph G is called *sane* if the following conditions are satisfied for every node x:

(a) the margin of x is nonempty;
(b) the subgraphs induced in G by the cone of x and by the component of x are connected;
(c) every vertex of the adhesion of x has a neighbor in the component of x.

Intuitively, saneness means that the decomposition respects the connectivity of subgraphs corresponding to its subtrees. Indeed, it is straightforward to see from the definition, that all the marginal graphs of a sane decomposition are nonempty and connected. A similar notion of *internal connectivity* was used by Lapoire [14]. The following lemma, which may be considered folklore, shows that any tree decomposition can be sanitized. The lemma is colored red because its proof is deferred to the full version, we use this convention throughout this extended abstract.

Lemma 2.8. *Suppose t is a tree decomposition of a graph G. Then there exists a sane tree decomposition s of G where every bag in s is a subset of some bag in t. In particular, if $\mathtt{tw}(G) \leq k$, then G admits a sane tree decomposition of width at most k.*

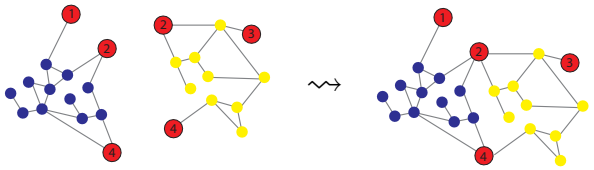
Proof of Theorem 2.4. The proof of Theorem 2.4 is a combination of Lemmas 2.5 and 2.6. This requires some technical care, but does not involve any substantially new ideas; a complete proof is in the full version of the paper.

2.3 Courcelle's conjecture

In this section we use Theorem 2.4 to prove the conjecture of Courcelle mentioned in the introduction. We use a syntax slightly different than Courcelle.

Definition 2.9. Define an *interface graph* to consist of an *arity* $k \in \mathbb{N}$, an *underlying graph* G called the *underlying graph* and an *interface mapping*, which is an injective partial function from $\{1, 2, \ldots, k\}$ to vertices of the underlying graph. If image of $i \in \{1, 2, \ldots, k\}$ under the interface mapping is defined, it is called the *i-th interface vertex*. Then i is the *name* of this interface vertex.

Interface graphs of arity k are called k-interface graphs. If $\mathbb{G}$ and $\mathbb{H}$ are k-interface graphs, then their *gluing* $\mathbb{G} \oplus \mathbb{H}$ is the k-interface graph defined as follows. The underlying graph is the disjoint union of the two underlying graphs, with the interface vertices having the same names in $\mathbb{G}$ and $\mathbb{H}$ being fused. In other words, for any name $i \in \{1, 2, \ldots, k\}$ that is used both in $\mathbb{G}$ and in $\mathbb{H}$ (i.e. is in the intersection of the domains of the interface mappings), we fuse the corresponding i-th interface vertices in $\mathbb{G} \oplus \mathbb{H}$. If this process creates any parallel edges, we remove the duplicates. The names of the interface vertices in $\mathbb{G} \oplus \mathbb{H}$ are inherited from the arguments. We illustrate this definition with the following example:



In Courcelle's syntax from [5], the gluing essentially corresponds to substituting $\mathbb{H}$ for the hyperedge consisting of the interface vertices inside $\mathbb{G}$.

Recognisability. Let Π be a property of graphs. We say that two k-interface graphs $\mathbb{G}_1$ and $\mathbb{G}_2$ are Π-equivalent if

$$\mathbb{G}_1 \oplus \mathbb{H} \text{ satisfies } \Pi \qquad \text{iff} \qquad \mathbb{G}_2 \oplus \mathbb{H} \text{ satisfies } \Pi.$$

holds for every k-interface graph $\mathbb{H}$. This is an equivalence relation on k-interface graphs. If there are finitely many equivalence classes, then we say that Π is *k-recognisable*. Finally, we say that Π is *recognisable* if it is k-recognisable for every k. This is equivalent to Definition 1.12 in [5].

Courcelle's conjecture. In what follows, we consider the logic *counting* MSO which is the extension of MSO on graphs by predicates of the form "the size of set X is divisible by m" for every constant m. The following result was stated as Conjecture 1 in [5].

Theorem 2.10. *If a property of graphs that have treewidth bounded by a constant is recognisable, then it is definable in counting* MSO.

As mentioned in the introduction, the converse implication was proved by Courcelle in [4]. In his later work [5], Courcelle proposed the following approach to proving Theorem 2.10. Call a property of graphs Π *strongly context-free* if (informally), given any graph G from Π, some constant-width decomposition of G can be nondeterministically defined in MSO. If we prove that the class of graphs of treewidth k is strongly context-free (which is stated as Conjecture 2 in [4]), then Theorem 2.10 would follow from the following lemma.

Lemma 2.11. *Let $k \in \mathbb{N}$ and let Π be a k-recognisable property of graphs. There is a formula of counting* MSO *over the vocabulary of tree decompositions which is true in exactly those structures which represent a tree decomposition of width k where the underlying graph satisfies Π.*

Lemma 2.11 is essentially proved in [5] (cf. Theorem 4.8 therein); for completeness we give a proof adjusted to our notation in the full version of the paper. The statement that the class of graphs of treewidth at most k is strongly context-free is, up to insignificant differences in definitions, equivalent to our Theorem 2.4. Hence, we can complete the proof of Theorem 2.10 as Courcelle suggested.

Proof of Theorem 2.10. Let Π be a property of graphs of treewidth at most k. Apply Theorem 2.4 to k, yielding an MSO transduction, which maps graphs of treewidth at most k to tree decompositions of width at most $f(k)$. Apply Lemma 2.11 to $f(k)$ and the property Π, yielding a formula of counting MSO which tests Π on tree decompositions of width at most $f(k)$. The result follows by using the Backwards Translation Theorem. □

We believe that a stronger statement holds, namely: if Π is a property of graphs of treewidth k, then already being k-recognisable implies definability in counting MSO. This claim would follow from the stronger version of Theorem 2.4 described after its statement.

3. Guidance systems

In this section, we introduce guidance systems. The definition is a variant of the guidance systems defined in [3]. Guidance systems are used in the proofs of Lemmas 2.5 and 2.6, which are found in Sections 4 and 5.

Definition 3.1. A *guidance system* Λ over a graph G is a family of in-trees (i.e. connected in-forests), where each in-tree is obtained by orienting the edges of some subgraph of G. For a vertex u, define

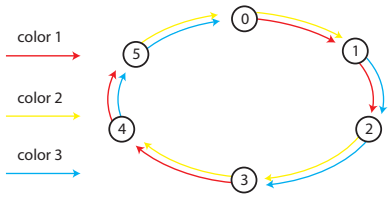
$$\Lambda(u) = \{v : \text{some tree from } \Lambda \text{ contains } u \text{ and has root } v\}.$$

A *coloring* of a guidance system is an assignment of trees to colors so that trees with the same color have disjoint vertex sets.

Example 3.2. Let G be an undirected cycle of length six, with vertices called $\{0, 1, \ldots, 5\}$, and edges being neighbor modulo 6. Consider the following guidance system

$$\Lambda = \{u \to u+1 \to u+2 : u \in \{0, \ldots, 5\}\}$$

where addition is modulo 6. This guidance system is 3-colorable:



For this guidance system,

$$\Lambda(u) = \{u, u+1, u+2 \mod 6\} \qquad \text{for } u \in \{0, \ldots, 6\}.$$

Guidance system are used to recognize sets of vertices in a graph, in the sense defined below.

Definition 3.3. Let G be a graph. A set of vertices X is said to be *captured* by a vertex u in a guidance system Λ if $X \subseteq \Lambda(u)$ holds. A family of sets of vertices is said to be captured by Λ if each set is captured by some vertex.

From adhesions to a tree decomposition. In Lemma 3.4 below, we show that in order to produce a sane tree decomposition with an MSO interpretation, it suffices to capture all its adhesions with a bounded number of colors. Note that in the lemma below we do not restrict the sizes of bags in the tree decompositions; e.g., a decomposition with all vertices in one bag has no adhesions and therefore falls into the scope of the lemma.

Lemma 3.4. *For every $k \in \mathbb{N}$ there is an* MSO *transduction from graphs to tree decompositions which maps every graph G to all sane tree decompositions of G whose family of adhesions can be captured by a k-colorable guidance system over G.*

The proof of Lemma 3.4 is actually quite non-trivial. We use the connectivity conditions given by saneness in order to be able to guess the bags of a sane tree decomposition. Also, instead of the original graph, we need to work with the graph obtained by turning all adhesions into cliques. This structure can be constructed by an MSO transduction which guesses a guidance system that captures all the adhesions. The fact that the guidance system can be colored using few colors is necessary for it to be guessable in MSO.

Stability under small modifications. In Lemma 3.5 below, we show that the number of colors needed to capture a family of sets by a guidance system is stable under removing or adding vertices. If G is a graph, we write $G - u$ for the subgraph induced by removing u from the vertices.

Lemma 3.5. *Let G be a graph, let $\mathcal{X}$ be a family of subsets of vertices, and let u be a vertex.*

(1) If every set in $\mathcal{X}$ is contained in some connected component of G and

$$\mathcal{X} - u \stackrel{def}{=} \{X - \{u\} : X \in \mathcal{X}\}$$

is captured by a k-colorable guidance system over G, then $\mathcal{X}$ is captured by a $(k+1)$-colorable guidance system over G.

(2) If every set from $\mathcal{X}$ is contained in some connected component of $G - u$ and $\mathcal{X}$ is captured by a k-colorable guidance system over G, then $\mathcal{X}$ is captured by a $2k$-colorable guidance system over $G - u$.

4. Graphs of bounded pathwidth

In this section we prove Lemma 2.5, which says that an MSO transduction can transform a graph of bounded pathwidth into a tree decomposition. Our proof relies on the guidance systems defined in the previous section. We first outline the plan. In Section 4.1, we define a graph parameter called guided treewidth. In Section 4.2 we show that bounded pathwidth implies bounded guided treewidth. A combination of this result with the fact that guidance system can be expressed in MSO (Lemma 3.4) yields Lemma 2.5.

4.1 Guided treewidth

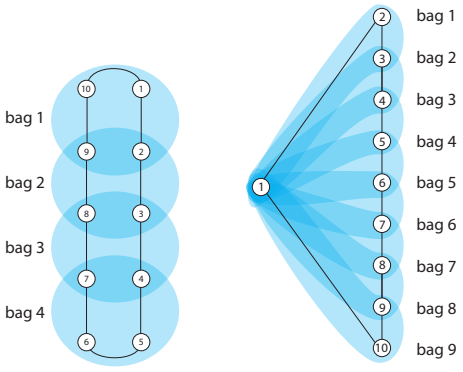
The following definition can be seen as a new graph parameter.

Definition 4.1. Define the *guided treewidth* of a graph G, denoted by $\mathtt{gtw}(G)$, to be the smallest k such that there exists a tree decomposition of G where all bags are captured by some k-colorable guidance system over G.

Note that if a bag is captured by a k-colorable guidance system, then it has size at most k. Hence, the guided treewidth of a graph is an upper bound on its treewidth. Since adhesions are contained in bags, tree decompositions whose bags are captured by a k-colorable guidance system fall into the scope of Lemma 3.4, and can be produced by an MSO transduction.

The goal of this section is to show that bounded pathwidth implies bounded guided treewidth, and therefore, by the above discussion, tree decompositions of graphs of bounded pathwidth can be produced by an MSO transduction. To illustrate guided treewidth, we show an example where a poorly chosen tree decomposition needs a large number of colors to be captured.

Example 4.2. Consider a cycle with vertices $\{1, \ldots, 2n\}$. For this cycle, consider a path decomposition with $n-1$ bags, where the i-th bag contains vertices i, $i+1$, $2n-i$, and $2n-i+1$. A picture for $n = 5$ is in the left panel of the figure below:



The path decomposition above has suboptimal width, but could be made optimal by adding a clique of size 4 to the graph. It is easy to see that any guidance system capturing this decomposition requires a number of colors that is linear in n, because such a guidance system has to contain a set of trees of linear size that all share one of the arcs (v_1, v_{2n}), (v_{2n}, v_1), (v_n, v_{n+1}), or (v_{n+1}, v_n).

To fix the problem we use a different path decomposition, with $2n-2$ bags, such that the u-th bag contains $\{1, u, u+1\}$. This decomposition is depicted in the right panel of the figure above. To capture its bags, we use a 3-colorable guidance system colorable. The first color is used to describe a directed path

$$2 \to 3 \to \cdots \to 2n \to 1.$$

The remaining two colors are used alternately to connect each vertex with its successor, similarly as in Example 3.2. Concluding, each cycle admits a tree (even path) decomposition that can be captured by a 3-colorable guidance system.

In the next section, we strengthen the result from the above example, and show that bounded pathwidth implies bounded guided treewidth. Before passing to the next section, we show that guided treewidth is robust with respect to graph operations like disjoint union (denoted $\uplus$) or adding/removing a single vertex. The proof is a simple application of Lemma 3.5.

Lemma 4.3. *Let G, G' be graphs and let u be a vertex in G. Then*

$$\mathtt{gtw}(G \uplus G') = \max(\mathtt{gtw}(G), \mathtt{gtw}(G')) \quad (1)$$

$$\mathtt{gtw}(G) \leq \mathtt{gtw}(G - u) + 1 \quad (2)$$

$$\mathtt{gtw}(G - u) \leq 2 \cdot \mathtt{gtw}(G) \quad (3)$$

4.2 Guided treewidth is bounded by pathwidth

We now state and prove the main result of Section 4, which is that bounded pathwidth implies bounded guided treewidth.

Lemma 4.4. *There exists a function $f(k) \in 2^{2^{\mathcal{O}(k^2)}}$ such that*

$$\mathtt{gtw}(G) \leq f(\mathtt{pw}(G)) \qquad \text{for every graph } G.$$

Note the asymmetry in the lemma: we assume bounded pathwidth, but produce a tree decomposition. It can be easily seen that Lemma 2.5 follows by combining Lemma 4.4 with Lemmas 2.8 and 3.4. A full argument is in the full version of the paper.

The rest of Section 4 is devoted to proving Lemma 4.4. In Section 4.2.1 we define bi-interface graphs, which give an alternative algebraic definition of pathwidth. Then, in Section 4.2.2, we use the Factorization Forest Theorem to prove Lemma 4.4.

4.2.1 Bi-interface graphs

Recall the interface graphs as defined in Section 2.3. In our approach to pathwidth, we use such an enriched version of this definition, where a graph is supplied with two sets of interfaces: left and right. Here is the formal definition.

Definition 4.5. A *bi-interface graph* consists of an *arity* $k \in \mathbb{N}$, an *underlying graph* G, and two partial injective functions left, right from $\{1, \ldots, k\}$ to the vertices of G. We use the name *i-th left interface* for $\mathsf{left}(i)$, likewise for the i-th right interface. Moreover, we require that if a vertex is simultaneously an i-th left interface and a j-th right interface, then $i = j$.

Thus, the interface names of a bi-interface graph of arity k are numbers between 1 and k, however not all of them need to be used.

In Section 2.3 we showed how to glue interface graphs. For bi-interface graphs we use a similar notion, which is probably best seen in a picture, see Figure 1. Here is the formal definition. Let $\mathbb{G}_1, \mathbb{G}_2$ be two bi-interface graphs of the same arity k. Define their gluing $\mathbb{G}_1 \oplus \mathbb{G}_2$ as follows. Take the disjoint union of the underlying graphs, and fuse the i-th right interface of $\mathbb{G}_1$ with the i-th left interface of $\mathbb{G}_2$, whenever both are defined. As before, remove the duplicates whenever any parallel edge is created in this operation. As the left interface function take the left interface function of $\mathbb{G}_1$, and as the

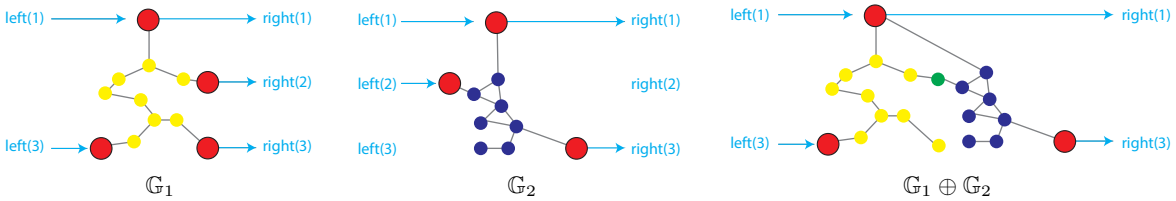


Figure 1: Two bi-interface graphs and their gluing. The interface nodes are red, the incoming/outgoing arrows indicate left/right interfaces, respectively. Some of the interfaces are undefined, e.g. the second left interface in $\mathbb{G}_1$, and the first left and right interfaces are equal.

right interface function take the right interface function of $\mathbb{G}_2$. It is easy to verify that if $\mathbb{G}_1$ and $\mathbb{G}_2$ were both bi-interface graphs of arity k, then $\mathbb{G}_1 \oplus \mathbb{G}_2$ is also a bi-interface graph of arity k. Note that in $\mathbb{G}_1 \oplus \mathbb{G}_2$ we forget the information about the right interfaces of $\mathbb{G}_1$ and the left interfaces of $\mathbb{G}_2$.

The gluing operation defined above is associative, turning the set of bi-interface graphs of arity k into a semigroup. A product

$$\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_n$$

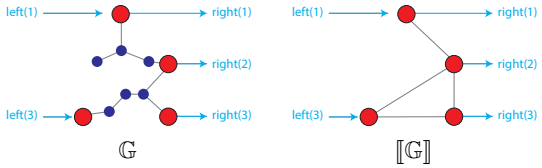
in this semigroup is essentially the same thing as a path decomposition, where the bags are the bi-interface graphs $\mathbb{G}_1, \ldots, \mathbb{G}_n$, and the interface functions say how the bags are connected. Hence the following lemma, whose straightforward proof is omitted.

Lemma 4.6. *A graph has pathwidth at most k if, and only if it is the underlying graph of a bi-interface graph of the form*

$$\mathbb{G}_1 \oplus \ldots \oplus \mathbb{G}_n$$

where each $\mathbb{G}_i$ has arity k and at most $k+1$ vertices.

Abstraction. Call two bi-interface graphs *isomorphic* if there is a bijection between their vertex sets that respects graph edges and the name of each interface. For a graph G and a subset of vertices X, by the *torso* of G with respect to X we mean a graph on vertex set X where two vertices are adjacent if they can be connected in G by a path whose internal vertices do not belong to X. Define the *abstraction* $[\![\mathbb{G}]\!]$ of a bi-interface graph $\mathbb{G}$ to be the isomorphism type of the bi-interface graph obtained by taking the torso of the underlying graph with respect to the interface vertices, and preserving the interface functions. Here is a picture of a bi-interface graph and its abstraction:



Define $\mathbb{A}_k$ to be the possible abstractions of bi-interface graphs of arity k. This is a finite set of size $2^{\mathcal{O}(k^2)}$. The abstraction function $\mathbb{G} \mapsto [\![\mathbb{G}]\!]$ is compositional in the sense that $[\![\mathbb{G}_1 \oplus \mathbb{G}_2]\!]$ is uniquely determined by $[\![\mathbb{G}_1]\!]$ and $[\![\mathbb{G}_2]\!]$. This means that $\mathbb{A}_k$ can be uniquely endowed with a multiplication operation which makes the abstraction function a semigroup homomorphism from bi-interface graphs of arity k to $\mathbb{A}_k$. From now on we will treat $\mathbb{A}_k$ as a semigroup, and the abstraction function as a homomorphism.

4.2.2 Simon's Factorization Forest Theorem

We now recall Simon's Factorization Forest Theorem from semigroup theory, whose application is the cornerstone of our proof of Lemma 2.5. Intuitively, the theorem says that if h is a homomorphism from words into a finite semigroup S, then every word can be recursively factorised until reaching single letters, with the depth of the factorisation depending only on the size of S, and each factorisation step respecting h in some way.

We write Σ^+ for the semigroup of nonempty finite words over alphabet Σ with concatenation. Let S be a finite semigroup and let

$$h : \Sigma^+ \to S$$

be a semigroup homomorphism. We define two types of factorizations for a word $u \in \Sigma^+$.

- **Binary**: a factorization into two factors
$$u = u_1 u_2.$$
- **Unranked**: a factorization into an arbitrary number of factors
$$u = u_1 \ldots u_n,$$
such that all factors $u_1, \ldots, u_n$ have the same image under h.

The *h-rank* of a word $u \in \Sigma^+$ is the natural number defined by induction on the length of u as follows. Single letters have rank 1. For a longer word u, its h-rank is

$$1 + \min_{u=u_1 \ldots u_n} \max_{i \in \{1,\ldots,n\}} h\text{-rank of } u_i$$

where the minimum ranges over factorizations (either binary or unranked) of u into nonempty factors. Using binary factorizations only, it is easy to see that every word has h-rank that is at most logarithmic in its length. Imre Simon proved that thanks to the unranked factorisations, one can achieve constant rank.

Theorem 4.7 (Factorisation Forest Theorem [15, 13])**.** *If $h\colon \Sigma^+ \to S$ is a semigroup homomorphism with S finite, then every word in Σ^+ has h-rank at most $3|S|$.*

The original result there is actually slightly stronger: in the unranked factorisations only idempotent values in the semigroup can be used. We will not use idempotence. The bound $3|S|$ is from [13], improving on the original exponential bound of Simon [15]. The word *forest* in the theorem's name is because the definition of rank implicitly uses a tree structure of factorisations.

We now return to the proof of Lemma 4.4, which says that guided treewidth is bounded by a function of the pathwidth. Consider the semigroup homomorphism

$$h : \Sigma^+ \to \mathbb{A}_k,$$

where Σ is the set of arity-k bi-interface graphs with at most $k+1$ vertices, and h is the homomorphism that glues a word to a single bi-interface graph and takes its abstraction. We will show that for every

$$\mathbb{G}_1 \cdots \mathbb{G}_n \in \Sigma^+$$

the guided treewidth of the corresponding glued graph is at most exponential in the h-rank of the word. More precisely:

$$\texttt{gtw}(\mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n) \leq 2^{\mathcal{O}(k \cdot (h\text{-rank}(\mathbb{G}_1 \cdots \mathbb{G}_n)))}. \tag{4}$$

By Lemma 4.6 and the Factorisation Forest Theorem, every graph of pathwidth k can be obtained from some word in Σ^+ with h-rank

bounded by $3|\mathbb{A}_k| = 2^{\mathcal{O}(k^2)}$, thus proving Lemma 4.4. It remains to show (4). For this, we use induction on the h-rank. The induction base follows from the observation that the guided treewidth of a graph is at most the number of its vertices, which follows directly from claim (2) of Lemma 4.3. For the induction step, we use the Binary and Unranked lemmas stated below.

Lemma 4.8 (Binary). *If $\mathbb{G}_1, \mathbb{G}_2$ are arity-k bi-interface graphs, then*

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

The Binary Lemma, used for binary factorizations, follows immediately from Lemma 4.3. In the proof, we observe that gluing two bi-interface graphs can be modelled as follows: (i) remove from both graphs the interface vertices that are going to be fused, (ii) take the disjoint union, and (ii) reintroduce the removed vertices.

For unranked factorisations, we use the following lemma.

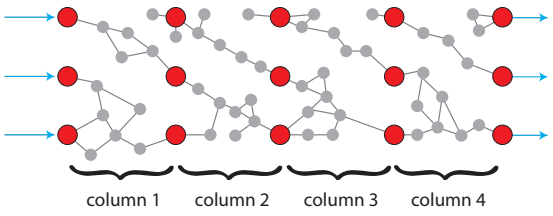
Lemma 4.9 (Unranked). *Let $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_n$ be bi-interface graphs of arity k which all have the same abstraction. Then*

$$\mathtt{gtw}(\mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n) \leq k(4k^2+5) + 8^k \cdot \max_{i \in \{1,\ldots,n\}} \mathtt{gtw}(\mathbb{G}_i).$$

The proof of the Unranked Lemma is more involved and presented in the following section.

4.2.3 Proof of the Unranked Lemma

We begin by introducing some notation. If $\mathbb{G}_1, \ldots, \mathbb{G}_n$ are bi-interface graphs of the same arity and $i \in \{1, \ldots, n\}$, then there is a natural injective mapping column_i that associates the vertices of $\mathbb{G}_i$ with the corresponding vertices of its copy in $\mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n$. Let us call this function the *i-th column function*, and its image the *i-th column*. Here is a picture that illustrates columns:



We begin by observing that if we glue many copies of the same bi-interface graph, then vertices in the same column can either be connected by a path visiting few columns, or not at all. Here, by a path *staying within* a set of columns we mean that the vertex set of the path is contained in the union of these columns.

Lemma 4.10. *Let $\mathbb{G}$ be a bi-interface graph. If two vertices of*

$$\mathbb{G}^n \stackrel{\text{def}}{=} \overbrace{\mathbb{G} \oplus \cdots \oplus \mathbb{G}}^{n \text{ times}}.$$

are in the same column $i \in \{1, \ldots, n\}$ and can be connected by a path, then they can also be connected by a path that stays only in columns j satisfying $|j - i| \leq k^2$, where k is the arity of $\mathbb{G}$.

We now show that if we glue many bi-interface graphs which are not necessarily equal but have the same abstraction, then there is a guidance system, colorable by a small number of colors, which takes each vertex to the reachable interfaces from its own column.

Lemma 4.11. *Let $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_n$ be bi-interface graphs with the same arity k and the same abstraction, and such that the left and right interfaces are disjoint. Consider the gluing*

$$\mathbb{G} = \mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n.$$

For $i \in \{1, \ldots, n\}$ and a vertex u in the i-th column, define $I_i(u)$ to be those interface vertices of the i-th column which are reachable from u by a path in $\mathbb{G}$. Then there is a guidance system Λ over $\mathbb{G}$, which can be colored by at most $4k(k^2+1)$ colors, such that

$$I_i(u) \subseteq \Lambda(u) \quad \text{for each } i \in \{1, \ldots, n\} \text{ and } u \text{ in the } i\text{-th column.}$$

The next step is to prove the Unranked Lemma in the special case where the bi-interface graphs have disjoint left and right interfaces. The proof is an application of Lemma 4.11, which ensures us that connections that have to be realized by the sought guidance system have a local character.

Lemma 4.12. *Let $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_n$ be bi-interface graphs of arity k which all have the same abstraction, and such that the left and right interfaces are disjoint. Then*

$$\mathtt{gtw}(\mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n) \leq 4k(k^2+1) + 4^k \cdot \max_{i \in \{1,\ldots,n\}} \mathtt{gtw}(\mathbb{G}_i).$$

We now finish the proof of the Unranked Lemma. Let

$$\mathbb{G} \stackrel{\text{def}}{=} \mathbb{G}_1 \oplus \cdots \oplus \mathbb{G}_n$$

be a gluing of bi-interface graphs of the same arity k and the same abstraction. Define U_i to be the vertices in $\mathbb{G}_i$ that are both left and right interfaces at the same time and define $\mathbb{H}_i$ to be $\mathbb{G}_i$ with U_i removed. Consider the gluing

$$\mathbb{H} \stackrel{\text{def}}{=} \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_n.$$

By the assumption that the abstractions of all of $\mathbb{G}_1, \ldots, \mathbb{G}_n$ are the same, it follows that the image of U_i under the i-th column function is the same set, independent of i, and this set has at most k vertices. Therefore, $\mathbb{H}$ is equal to $\mathbb{G}$ with at most k vertices removed. By claim (2) of Lemma 3.5 we infer that

$$\mathtt{gtw}(\mathbb{G}) \leq \mathtt{gtw}(\mathbb{H}) + k. \tag{5}$$

The left and right interfaces are disjoint in each $\mathbb{H}_i$, and these graphs also have the same abstraction, so we can use Lemma 4.12 to get:

$$\mathtt{gtw}(\mathbb{H}) \leq 4k(k^2+1) + 4^k \cdot \max_{i \in \{1,\ldots,n\}} \mathtt{gtw}(\mathbb{H}_i). \tag{6}$$

Finally, each $\mathbb{H}_i$ is obtained from $\mathbb{G}_i$ by removing at most k vertices, and hence we can apply claim (3) of Lemma 3.5 to infer that

$$\mathtt{gtw}(\mathbb{H}_i) \leq 2^k \cdot \mathtt{gtw}(\mathbb{G}_i). \tag{7}$$

By combining (5), (6), and (7), we obtain the desired upper bound on the guided treewidth of $\mathbb{G}$.

5. Graphs of bounded treewidth

In this section we prove Lemma 2.6. Our strategy is to show the following result on guidance systems.

Lemma 5.1. *Let G be a graph of treewidth at most k. Then G admits a tree decomposition t° with the following properties:*

(a) every marginal graph has pathwidth at most $2k+1$;
(b) the family of adhesions of t° can be captured by a guidance system colorable with $4k^3 + 2k$ colors.

From the lemma above, we can deduce Lemma 2.6 as follows. Take interpretation $\mathcal{S}_{4k^3+2k}$ from Lemma 3.4. Lemma 5.1 implies that if the treewidth of the input graph is at most k, then at least one output of $\mathcal{S}_{4k^3+2k}$ satisfies the conditions of Lemma 2.6.

The rest of Section 5 is devoted to proving Lemma 5.1. In Section 5.1, we prove a "local" variant of the lemma, which provides one step of the construction. In Section 5.2, we iterate the local variant to construct a global decomposition, thus proving Lemma 5.1.

5.1 A local variant of Lemma 2.6

In this section, we state and prove Lemma 5.2, which can be seen as a local variant of Lemma 5.1. We begin with some hypergraph terminology, which is used in its statement and proof.

Hypergraphs. A *hypergraph* consists of a set of vertices together with a multiset of nonempty subsets of the vertices, called the *hyperedges*. Note the use of multisets: hyperedges can appear multiple times. If H is a hypergraph, the hypergraph *induced* by a subset of vertices X, denoted by $H[X]$, is the hypergraph where the vertices are X and the hyperedges are intersections of the original hyperedges with X, with the empty ones removed. A *path* in a hypergraph is a sequence

$$(u_1, e_1, u_2, \ldots, u_p, e_p, u_{p+1}),$$

where u_i are pairwise different vertices, e_i are pairwise different edges, and vertices u_i, u_{i+1} are contained in hyperedge e_i for each $i = 1, 2, \ldots, p$. Vertices u_1 and u_{p+1} are the *endpoints*, and the path is said to *go* from u_1 to u_{p+1}. Each vertex u_i and hyperedge e_i is said to be *traversed* by the path. Connected components in a hypergraph are defined by path-connectedness in a natural manner. Tree decompositions of hypergraphs are defined as for graphs, except that every hyperedge must be contained in some bag.

Prefixes and their torsos. Let t be a sane tree decomposition of a graph G. A *prefix* of t is a set of nodes Z in t that is closed under taking ancestors. If Z is a prefix, define ∂Z to be the nodes of t that are not in Z, but their parent is in Z. For a prefix Z, define

$$\mathsf{hypertorso}(t, Z)$$

to be the hypergraph obtained by taking the subgraph of G induced by the union of the bags in Z, and then for every $z \in \partial Z$ a hyperedge for the adhesion of z. When adding hyperedges, we respect multiplicities, i.e. if n nodes in ∂Z have the same adhesion, then this adhesion is used n times in $\mathsf{hypertorso}(t, Z)$.

We are now ready to state the local version of Lemma 5.1.

Lemma 5.2. *Let t be a width k sane decomposition of a connected graph G. Let u, v be vertices in the root bag (note that there is a unique root due to connectedness). Then there exists a nonempty prefix Z of t with the following properties:*

(a) the pathwidth of $\mathsf{hypertorso}(t, Z)$ is at most $2k + 1$, and
(b) the vertices u and v can be connected by two paths in $\mathsf{hypertorso}(t, Z)$ such that if a hyperedge is traversed by both paths, then it is an edge of G.

The rest of Section 5.1 is devoted to proving the above lemma. The proof uses a Menger style argument, so we begin by defining terminology for hypergraph networks.

Networks. Define a *network* to be a connected hypergraph together with two distinguished different vertices, called the *source* and the *sink*. We extend all notation from hypergraphs to networks in a natural manner. A *cutedge* in a network is a hyperedge which appears in every path from the source to the sink; equivalently, the removal of a cutedge makes the source and the sink fall into different connected components. The following claim is straightforward.

Lemma 5.3. *In every network one can order all of the cutedges into a sequence $(e_1, \ldots, e_p)$ such that every path from the source to the sink visits the cutedges in this order.*

The sequence $(e_1, \ldots, e_p)$ yielded by Lemma 5.3 will be called the *cutedge sequence* of a network. Define a *cutedge component* in a network to be a connected component of the hypergraph obtained by removing the cutedges. The following claim is straightforward.

Lemma 5.4. *Consider a network. Let $(e_1, \ldots, e_p)$ be its cutedge sequence, and define e_0, e_{p+1} be the singletons of the source and the sink, respectively. Then every cutedge component intersects exactly one or exactly two among elements $e_0, e_1, \ldots, e_p, e_{p+1}$. In the former case, the intersected e_i is a cutedge; i.e. i is not equal to 0 or $p + 1$. In the latter case, the two intersected elements must be consecutive in the sequence.*

The above lemma motivates the following terminology for a cutedge component. If it intersects two consecutive elements e_i and e_{i+1} in the cutedge sequence extended by the singletons of the source and the sink, then it is called an (e_i, e_{i+1})-*bridge*. If a cutedge component is not a bridge of any kind, and therefore it intersects exactly one cutedge e_i, then it is called an e_i-*appendix*. This terminology is illustrated in Figure 2.

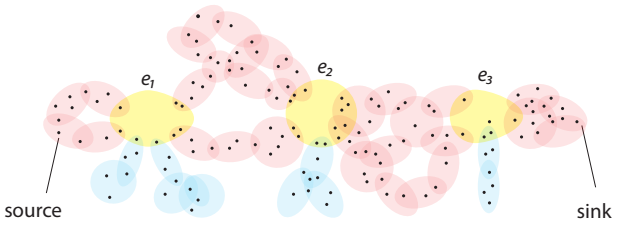


Figure 2: A network. The yellow hyperedges are the cutedges. The red hyperedges are those that contain bridges, the blue ones are those that contain appendices. Note the vertex in e_3 which participates in no blue or red hyperedges, this vertex is a singleton e_3-appendix.

The following lemma is proved by applying Menger's theorem, for every $i \in \{0, 1, \ldots, p\}$, to the union of all (e_i, e_{i+1})-bridges.

Lemma 5.5. *In every network one can find two paths from the source to the sink, such that a hyperedge is traversed by both paths if, and only if it is a cutedge.*

The invariant. To prove Lemma 5.2, we will start with a prefix of the decomposition that contains only the root, and keep on extending the prefix until it satisfies condition (b). While extending the prefix, we will preserve an invariant which implies condition (a). We now describe the invariant. Let t, u, v be as in Lemma 5.2. For a prefix Z of t, consider the network obtained from $\mathsf{hypertorso}(t, Z)$ by choosing u as the source and v as the sink. We will maintain the invariant that this network is k-thin in the sense defined below.

Definition 5.6. Consider a network with hypergraph H and cutedge sequence $(e_1, \ldots, e_p)$; we follow the convention that e_0, e_{p+1} denote the singletons of the source and the sink, respectively. Define V_i to be the union of the vertex sets of all (e_i, e_{i+1})-bridges, for $i = 0, 1, \ldots, p$, and W_i to be the union of the vertex sets of all e_i-appendices, for $i = 1, \ldots, p$. The network is called k-thin if:

(a) for every $i \in \{0, 1, \ldots, p\}$, the induced hypergraph $H[V_i]$ admits a path decomposition of width at most $2k + 1$ where the first bag contains $e_i \cap V_i$ and the last bag contains $e_{i+1} \cap V_i$;
(b) for every $i \in \{1, 2, \ldots, p\}$, the induced hypergraph $H[W_i]$ admits a path decomposition of width at most k where the first bag contains $e_i \cap W_i$.

The following lemma shows that our invariant implies condition (a) in Lemma 5.2. The proof is a simple surgery on decompositions certifying thinness.

Lemma 5.7. *A k-thin network has pathwidth at most $2k + 1$.*

Before finishing the proof of Lemma 5.2, we show that thinness is preserved under a certain kind of replacements. Let H, K be hypergraphs such that the intersection of their vertex sets is equal to a hyperedge e of H. Define $H[e \to K]$ to be the following hypergraph. The vertex set is the union of the vertex sets in H, K. The hyperedges are the multiset union of the hyperedges in H, K, with the hyperedge e removed. The following lemma shows that the above defined replacement preserves k-thinness of networks, assuming that e is a cutedge and K is small. The proof is a technical, though conceptually simple analysis of the relationship between cutedges before and after the replacement.

Lemma 5.8. *Consider a k-thin network with hypergraph H. Let K be a connected hypergraph with at most $k+1$ vertices, with no hyperedge larger than k, and such that the intersection of the vertices of H and K is equal to some cutedge e of H. Then the hypergraph $H[e \to K]$, with the same source and sink as in H, is also a k-thin network.*

We are now ready to prove Lemma 5.2.

Proof. For a prefix Z of the tree decomposition t, define the *network of Z* to be the network obtained from $\mathsf{hypertorso}(t, Z)$ by choosing the source to be u and the sink to be v. This is indeed a network: the underlying hypergraph is connected because G itself is connected.

Initially, choose Z to be the prefix that contains only the root node of t. We will maintain the invariant that the network of Z is k-thin. The invariant is clearly satisfied by the initial choice, because the root bag has size at most $k+1$, and adhesions have sizes at most k (due to the saneness of t).

By Lemma 5.7, the invariant implies condition (a). We show below that if Z is a prefix satisfying the invariant, then either it satisfies condition (b), in which case we are done, or one can add a node to the prefix while maintaining the invariant. Since the tree decomposition t has a finite number of nodes, this process has to stop at some moment, thus proving the lemma.

Let then Z be a prefix such that the network of Z is k-thin. Apply Lemma 5.5, yielding two paths from the source to the sink in the network of Z, such that the only hyperedges traversed by both paths are the cutedges of the network of Z. If all these cutedges are original edges of G, then we are done, because Z satisfies condition (b). Otherwise, there is some cutedge e in the network of Z that is not an edge of G. By definition of the network of Z, the cutedge e corresponds to the adhesion of some $z \in \partial Z$. Again by definition, the network of $Z \cup \{z\}$ is obtained from the network of Z by: adding the bag of z to the vertices, removing the hyperedge e, and adding a hyperedge for every adhesion of a child of z. We now verify that this process is a special case of the replacement in Lemma 5.8. Indeed, if we define hypergraph $K = \mathsf{hypertorso}(t_z, \{z\})$, where t_z is the subtree of t rooted at z, then the network of $Z \cup \{z\}$ is obtained from the network of Z by replacing e by K. Observe that K has at most $k+1$ vertices, has no hyperedge larger than k due to the saneness of t, and is connected, again due to the saneness of t. Hence Lemma 5.8 ensures us that the network of $Z \cup \{z\}$ is also k-thin. □

5.2 Decomposition into low pathwidth parts

In this section we finish the proof of Lemma 5.1. We heavily use the notation from Definition 2.2.

Consider a tree decomposition t. For a distinguished set X of nodes in the decomposition t, which is required to include all the roots of t, we define a new tree decomposition $t/_X$ of the same underlying graph as follows. The nodes of $t/_X$ are X. For any node of t that is not in X, assign it to its closest ancestor that belongs to X, i.e., the ancestor from X for which there is no other node from X on the unique path between the node and the ancestor. Then the $t/_X$-bag of a node $x \in X$ is the union of the t-bags of the nodes assigned to x, plus the t-bag of x itself.

Lemma 5.1 will be obtained by taking any sane tree decomposition, and applying the following lemma to every connected component. Condition (a) of Lemma 5.1 will follow from Lemma 5.9(a), while condition (b) of Lemma 5.1 will follow from Lemma 5.10 proved at the end of this section.

Lemma 5.9. *Let t be a width k sane tree decomposition of a connected graph G. Then one can find a set of nodes X in t, which includes the root of t, and families of paths $\{\mathcal{P}_x\}_{x \in X}$, such that every $x \in X$ satisfies:*

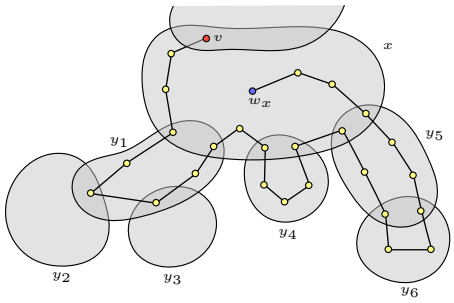


Figure 3: Example path in $\mathcal{P}_x$. This path contributes to the loads of nodes y_1, y_4, y_5, and y_6, but not of x, y_2, and y_3.

(a) The $t/_X$-marginal graph of x has pathwidth at most $2k+1$.
(b) Every element of $\mathcal{P}_x$ is a path in G that satisfies:
  (i) except for its endpoints, the path visits only vertices from the t-component of x;
  (ii) if $y \in X$ is a strict descendant of x, then restricting the path to the t-component of y yields an interval in the path.
(c) All paths in $\mathcal{P}_x$ have the same source, which belongs to the t-margin of x. Conversely, each vertex of the t-adhesion of x is a target of some path from $\mathcal{P}_x$.
(d) The following set of paths has size at most $2k^3$:

$$\mathrm{load}_x \stackrel{def}{=} \{P \in \mathcal{P}_y : y \in X \text{ is a strict ancestor of } x \text{ and } P \text{ intersects the } t\text{-component of } x\}.$$

Proof. Figure 3 illustrates the notions used in the lemma. We prove the following strengthening of the lemma, with sufficient parameters to be proved by induction.

($\star$) Let x_0 be a node of t, let I be a set of size at most $2k^3$, and let

$$\{(u_i, v_i)\}_{i \in I}$$

be a set of node pairs from the adhesion in x_0, with possible repetitions. One can find a set $X \ni x_0$ of nodes in the subtree of x_0, sets $\{\mathcal{P}_x\}_{x \in X}$ and a set of paths $\{Q_i\}_{i \in I}$ such that

(A) for every $i \in I$, the path Q_i goes from u_i to v_i and satisfies conditions (b:i) and (b:ii) in the lemma with respect to x_0;
(B) every $x \in X$ satisfies conditions (a)-(c) in the lemma and the following variant of (d): the size of load_x is at most

$$2k^3 - |\{i \in I : \text{path } Q_i \text{ intersects the } t\text{-component of } x\}|.$$

The lemma is a special case of ($\star$), by choosing x_0 to be the root of the tree decomposition t, and choosing I to be empty. It remains to prove ($\star$), which is done by induction on the number of nodes in the subtree of x_0.

Let x_0 and $\{(u_i, v_i)\}_{i \in I}$ be as in ($\star$). Choose (u, v) so that

$$I_0 = \{i \in I : (u_i, v_i) = (u, v)\}$$

has maximum size. If I is empty, choose I_0 to be empty and leave (u, v) undefined, this corner case will be considered separately. Since each candidate for (u, v) is in the adhesion of x_0, and adhesions have size bounded by k (due to the saneness of t), it follows that I_0 has size at least $|I|/k^2$.

Define t_0 to be the subtree of t rooted in x_0. Let t_0' be obtained from t_0 by removing from each bag those vertices of the adhesion of x_0 that are neither u nor v. It is easy to see that both t_0 and t_0' are sane tree decompositions. If u, v are defined, apply Lemma 5.2 to t_0' with distinguished vertices u, v, yielding a prefix Z and two paths in the hypergraph $\mathsf{hypertorso}(t_0', Z)$; call these paths P^1 and P^2. Since t_0 and t_0' have the same nodes, and these nodes are a

subset of the nodes of t, we can view Z as a connected set of nodes in each of these tree decompositions. In case I is empty and (u, v) is undefined, we choose Z to be the singleton of the root of t.

Let J be the disjoint union of I and the t-adhesion of x_0. For $i \in J$, define a path R_i in $\mathsf{hypertorso}(t_0, Z)$ as follows:

- For $i \in I - I_0$, define R_i to be a path from u_i to v_i, which does not visit the t-adhesion of x_0 except for its endpoints. Such a path exists by the saneness of t.
- Split the set I_0 into two parts, with the first part having half the size (rounded up). For i in the first part, define R_i to be P^1, and for i in the second part, define R_i to be P^2.
- For the remaining i, i.e. those from the t-adhesion of x_0, do the following. Independently of i, choose some vertex w in the t-margin of x_0, which is the same as the t_0-margin of x_0. This is possible because margins are nonempty in a sane decomposition. For each i in the adhesion of x_0, define R_i to be a path from w to i, which does not visit the adhesion of x_0 except for its endpoints. Such a path exists by the saneness of t.

By definition, every hyperedge in $\mathsf{hypertorso}(t_0, Z)$ is an edge of G or corresponds to an adhesion of some $z \in \partial Z$. Let $z \in \partial Z$. Define I^z to be those $i \in J$ for which path R_i uses the hyperedge corresponding to the adhesion of z. The following claim is the key step in this lemma, allowing us to apply the induction assumption.

Claim 1. *For every $z \in \partial Z$, the set I^z has size at most $2k^3$.*

The informal reason for the claim is that we have used the two paths P^1 and P^2, and at most one of them visits the hyperedge corresponding to the adhesion of z. Since I_0 constitutes a $1/k^2$-fraction of I, and I_0 is split in halves with respect to using P^1 or P^2, we see that around $1/2k^2$-fraction of paths R_i for $i \in I$ avoid the adhesion of z. This is enough to amortize for the new paths R_i for i from the t-adhesion of x_0.

For $i \in I^z$, let u_i^z be the vertex used by the path R_i immediately before the hyperedge corresponding to the adhesion of z, and let v_i^z be the vertex used immediately after. Take the vertex z, which is a proper descendant of x_0, and the family

$$\{(u_i^z, v_i^z)\}_{i \in I^z}.$$

and apply to it the induction assumption of $(\star)$, yielding

$$X^z \qquad \{\mathcal{P}_x^z\}_{x \in X^z} \qquad \{Q_i^z\}_{i \in I^z}.$$

For $i \in J$, define Q_i to be following path in G. Take the path R_i, which might use hyperedges corresponding to adhesions from ∂Z, and for every $z \in \partial Z$ replace the hyperedge corresponding to the adhesion of z, if it is used, by the path Q_i^z. The following claim follows directly from the construction and the induction assumption.

Claim 2. *For every $i \in J$, the path Q_i has the same source and target as R_i and satisfies conditions (b:i) and (b:ii) from the lemma.*

We now complete the proof of $(\star)$. Define

$$X = \{x_0\} \cup \bigcup_{z \in \partial Z} X^z.$$

Note how the above union is actually a partition. Define

$$\mathcal{P}_x = \begin{cases} \{Q_i : i \text{ in the } t\text{-adhesion of } x_0\} & \text{when } x = x_0 \\ \mathcal{P}_x^z & \text{when } x \in X^z. \end{cases}$$

Finally, define $\{Q_i\}_{i \in I}$ to be the restriction of the previously defined family $\{Q_i\}_{i \in J}$ to the smaller indexing set $I \subseteq J$.

Let us check that conditions (A) and (B) in the conclusion of $(\star)$ are satisfied by the above choices. Condition (A) is satisfied thanks to Claim 2. For $x \in X - \{x_0\}$, we check condition (B) in the following claim, which follows from the induction assumption.

Claim 3. *Condition (B) in $(\star)$ holds for each $x \in X - \{x_0\}$.*

It remains to check condition (B) for x_0, i.e. to check that x_0 satisfies conditions (a)-(c) and the variant of (d). For (a), we observe that every path decomposition for $\mathsf{hypertorso}(t_0', Z)$ is also a path decomposition of $t/_X$-margin of x_0, and therefore the latter graph has pathwidth at most $2k + 1$, by the conclusions of Lemma 5.2. (In the corner case when I was empty, the $t/_X$-margin of x_0 is the same as the t-margin, and therefore has size at most k.) For (b), we use Claim 2. Condition (c) follows by construction. Finally, condition (b) holds vacuously, because x_0 has no strict ancestors in X. □

To finish the proof of Lemma 5.1, we take $t^\circ = t/_X$ for X yielded by Lemma 5.9, and we need to check that the adhesions of t° can be captured by a guidance system that uses few colors. This requires a careful surgery on the paths families $\{\mathcal{P}_x\}_{x \in X}$ provided by the lemma. In particular, condition (d) is vital for the bounded on the number of colors used.

Lemma 5.10. *Let t and X be as in Lemma 5.9. The adhesions of the tree decomposition $t/_X$ can be captured by a guidance system that is colorable with $4k^3 + 2k$ colors.*

References

[1] H. L. Bodlaender, P. Heggernes, and J. A. Telle. Recognizability equals definability for graphs of bounded treewidth and bounded chordality. *Electronic Notes in Discrete Mathematics*, 49:559–568, 2015.

[2] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[3] M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. *Logical Methods in Computer Science*, 8(1), 2012.

[4] B. Courcelle. The Monadic Second-Order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[5] B. Courcelle. The Monadic Second-Order logic of graphs V: On closing the gap between definability and recognizability. *Theor. Comput. Sci.*, 80(2):153–202, 1991.

[6] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[7] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[8] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[9] M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.*, 44(1):114–159, 2015.

[10] L. Jaffke and H. L. Bodlaender. Definability equals recognizability for k-outerplanar graphs. In *IPEC'15*, volume 43 of *LIPIcs*, pages 175–186, 2015.

[11] V. Kabanets. Recognizability equals definability for partial k-paths. In *ICALP'97*, volume 1256 of *LNCS*, pages 805–815. Springer, 1997.

[12] D. Kaller. Definability equals recognizability of partial 3-trees and k-connected partial k-trees. *Algorithmica*, 27(3):348–381, 2000.

[13] M. Kufleitner. The height of factorization forests. In *MFCS 2008*, volume 5162 of *LNCS*, pages 443–454. Springer, 2008.

[14] D. Lapoire. Recognizability equals Monadic Second-Order definability for sets of graphs of bounded tree-width. In *STACS'98*, volume 1373 of *LNCS*, pages 618–628. Springer, 1998.

[15] I. Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990.