A Finite Presentation of Graphs of Treewidth at Most Three

Amina Doumane
Plume, LIP, CNRS, ENS de Lyon, France

Samuel Humeau
Plume, LIP, CNRS, ENS de Lyon, France

Damien Pous
Plume, LIP, CNRS, ENS de Lyon, France

Abstract

We provide a finite equational presentation of graphs of treewidth at most three, solving an instance of an open problem by Courcelle and Engelfriet.

We use a syntax generalising series-parallel expressions, denoting graphs with a small interface. We introduce appropriate notions of connectivity for such graphs (components, cutvertices, separation pairs). We use those concepts to analyse the structure of graphs of treewidth at most three, showing how they can be decomposed recursively, first canonically into connected parallel components, and then non-deterministically. The main difficulty consists in showing that all non-deterministic choices can be related using only finitely many equational axioms.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Mathematics of computing → Paths and connectivity problems

Keywords and phrases Graphs, treewidth, connectedness, axiomatisation, series-parallel expressions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.135

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://hal.science/hal-04560570` [10]

Supplementary Material *Software*: `https://perso.ens-lyon.fr/damien.pous/hypergraph/` [12] archived at `swh:1:dir:028863b2f75dde258591611a6c7c165e289db890`

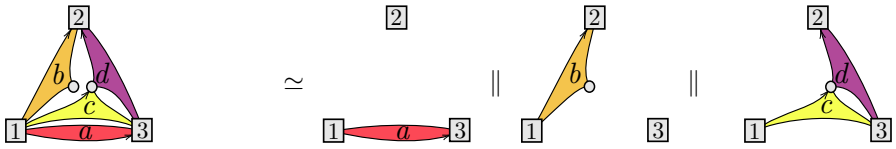
Funding This work was supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX- 0007) operated by the French National Research Agency (ANR).

Acknowledgements The second author would like to thank Ugo Giocanti for several discussions about graph minors and connectivity.

1 Introduction

Treewidth is a graph parameter measuring how close a graph is from a forest. It is frequently encountered in parameterised complexity: many NP-complete problems become polynomial or even linear once parameterised using treewidth [4]. This parameter admits many equivalent definitions. It was discovered at least by Bertelè and Brioschi [3], Halin [11], and Robertson and Seymour [13] via tree decompositions for their celebrated graph minor theorem. It was subsequently characterised using k-trees [2], k-elimination graphs [17, 16, 15], and chordal graphs [13].

One may also consider syntaxes, most often generalising series-parallel expressions [1, 8, 6]. There, the idea is to use terms to denote graphs, and an important problem is to understand when two terms denote the same graph. To this end, Courcelle and Engelfriet gave an



$\mathrm{lf}b \parallel \mathrm{f}\big((24)\mathrm{l}(c \parallel (23)\mathrm{l}a) \parallel (14)\mathrm{l}d\big) \equiv (23)\mathrm{l}a \parallel \mathrm{lf}b \parallel \mathrm{f}\big((24)\mathrm{l}c \parallel (14)\mathrm{l}d\big)$

Figure 1 Two parsings of a given graph.

equational axiomatisation for arbitrary graphs [7, p. 117]: a structured set of equations on terms from which it is possible to equate all terms denoting a given graph. Unfortunately, this axiomatisation is infinite. They note how finite fragments of the syntax make it possible to capture precisely the graphs up to a given treewidth, while finite restrictions of their axiomatisation seem to be incomplete.

This brings them to the following question [7, p. 118]: for a given bound k, is there a finite equational presentation of graphs of treewidth at most k? The case of $k = 1$ concerns forests and is relatively easy. The case of $k = 2$ has been given a solution a few years ago [5, 9]. Here we give a positive answer for $k = 3$: we provide a syntax of terms with an interpretation map g from terms to graphs whose image is exactly the set of graphs of treewidth at most three, and we give a finite list of equations whose equational theory ($\equiv$) characterises graph isomorphism ($\simeq$). In symbols, we prove that for all terms t, u,

$$g(t) \simeq g(u) \quad \text{if and only if} \quad t \equiv u$$

Like Arnborg, Courcelle, Proskurowski and Seese [1], we work with hypergraphs with a list of designated vertices, the *sources*, used as an interface to perform the following operations:

- *parallel composition* ($\parallel$): glue the graphs together along their sources.
- *permutation* (pG): given a permutation p, reorder the sources of G according to p.
- *lift* ($\mathrm{l}G$): add an isolated vertex to G and append it as last source.
- *forget* ($\mathrm{f}G$): remove the last source of G (keeping it as a mere vertex of the graph).

Consider the four graphs depicted in Figure 1, each with three sources denoted with numbered squares. The neighbours of each edge are ordered, which we indicate by drawing an arrow from the first to the second neighbour. The first graph on the left is the parallel composition of the three other ones. The second one can be obtained from a binary edge a (with interface its endpoints) by applying a lift to add a third isolated source and then swapping the last two sources: $(23)\mathrm{l}a$. The third one can be obtained from a ternary edge b (again with interface its endpoints), by forgetting the third source and adding a fresh one via a lift: $\mathrm{lf}b$. The last one can be constructed as $\mathrm{f}\big((24)\mathrm{l}c \parallel (14)\mathrm{l}d\big)$: reasoning top-down, we promote the inner vertex as a fourth source, and then we put in parallel two graphs each with a single ternary edge connecting three out of the four sources – both being obtained as appropriate permutations of lifted edges.

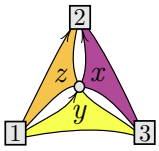
We call *parsings* the expressions we obtain when decomposing graphs as above.

The way we parsed the graph on the right generalises to all graphs: first promote all inner vertices as sources, and then build a large parallel composition of appropriately permuted and lifted edges. For instance, the first graph on the left also admits the following parsing:

$$\mathrm{ff}\big((23)\mathrm{lll}a \parallel (35)\mathrm{ll}b \parallel (24)\mathrm{ll}c \parallel (124)\mathrm{ll}d\big)$$

While such a parsing exists for all graphs, it goes through an intermediate graph with a large interface. Instead, the graphs of treewidth at most k are exactly those for which we can find a parsing where all intermediate graphs have at most $k+1$ sources (Proposition 4.1, [1, Proposition 4.1]). This syntax for denoting precisely the graphs of treewidth at most k is the starting point for the present work.

A derived operation is of particular interest. Consider three graphs x, y, z each with three sources, and combine them as depicted on the right to obtain a new graph $\circ(x,y,z)$. This operation can be defined from the previous ones using the expression below.



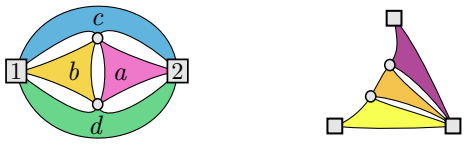
$$\circ(x,y,z) \triangleq \mathrm{f}\big((14)\mathrm{l}x \parallel (24)\mathrm{l}y \parallel (34)\mathrm{l}z\big)$$

It can be generalised into a n-ary operation on graphs with n sources, and when $n=2$ we recover the usual notion of *series composition* (with its arguments reversed).

Our goal is then to understand which laws are satisfied by the previous operations. Amongst the natural ones, we have that parallel composition is associative and commutative, that permutations commute over parallel compositions, and that applying two permutations in a row amounts to applying the composite permutation. There are more involved ones. For instance, we may also parse the first graph of Figure 1 by keeping edges a and c together as long as possible, resulting in the expression written below it. We thus have two rather different parsings for the same graph, which our axiomatisation should equate.

We proceed in two steps. First we use connectivity arguments in order to decompose any graph into a parallel composition of permuted lifts of *full prime graphs*: non-empty graphs which are connected through paths not using sources except possibly at their endpoints. Figure 1 actually provided an example of such a decomposition, which is always unique for a given graph. Using a few natural axioms, we show that every term can be rewritten under such a form (Proposition 5.1). This makes it possible to focus on full prime graphs in the second step, which is where the main difficulties arise.

A key property of full prime graphs of bounded treewidth is that either they are *atomic* (i.e., reduced to a permutation of an edge), or they have a *forget point*: at least one of their inner vertices can be promoted into a source without increasing the treewidth, thus making it possible to parse the graph as a forget operation. The difficulty is that forget points are not unique, resulting in several ways of parsing non-atomic full prime graphs. Consider for instance the following tetrahedron on the left, with only two vertices marked as sources.



Each of the two inner vertices is a forget point, so that modulo appropriate permutations of edges, we may parse this graph as $\mathrm{f}(\circ(a,b,c) \parallel d)$ or as $\mathrm{f}(\circ(a,b,d) \parallel c)$. In this case, the two forget points are relatively close, and one of our axioms makes it possible to jump directly from one parsing to the other. A similar situation arises with the graph given on the right.

The cornerstone of our completeness proof is the fact that two parsings of a non-atomic full prime graph can always be rewritten so as to agree on their forget point. This is Lemma 5.4, and most of the paper is devoted to proving it.

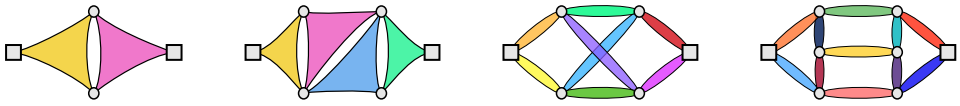


Figure 2 Separation pairs and graphs with several separation pairs.

To do so, we first delimit a class of vertices which we call *anchors*. Those can be thought of as a generalisation of cutvertices [8, Section 1.4]. When a full prime graph has an anchor, we show that we can use it as a universal agreement point (Lemma 7.1).

Unfortunately, there are graphs without anchors. We call them *hard*, they can be thought of as specific unseparable graphs [8, Section 1.4]. The rest of the proof consists in a structural analysis of hard graphs of treewidth at most three. We show that they admit *separation pairs*: every hard graph has the shape on the left of Figure 2, and every parsing can be rewritten as a double forget on some of these separation pairs (Lemma 8.5). However, as illustrated on the right, separation pairs are not unique. In order to finish the proof, we analyse how separation pairs relate to each other. There are easy cases like in the second graph of Figure 2 where the diagonal separation pair connects the two vertical ones, and previously encountered axioms make it possible to conclude. By analysing the shape of graphs excluding the triangle between their sources as a minor (Proposition 8.7 – note that cycles may still appear in such graphs), we show that all other situations reduce to one of the two graphs on the right of Figure 2. There, the two outer-vertical pairs of inner vertices are the only separation pairs, and they are disjoint. They correspond to distinct parsings of the same graph, and we must include the corresponding equations to complete our list of axioms.

For the sake of readability, some proofs and details are provided in the appendix of the full version [10].

2 Graphs, treewidth

A *ranked set* (or *signature*) is a set where every element has an associated natural number called its *arity*. Throughout the paper we fix a ranked set Σ of *letters*, which we call the *alphabet*. Given a set V, we denote by $\mathcal{L}(V)$ the ranked set of duplicate-free lists over V, where the arity of a list is its length.

We consider labelled and ordered hypergraphs with interfaces, defined as follows:

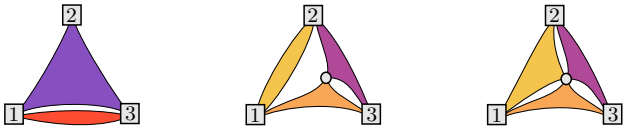
▶ **Definition 2.1.** *A* graph *is a tuple* $\langle V, E, n, l, i\rangle$ *where* V *is a finite set of* vertices*,* E *is a finite ranked set of* edges*,* $i \in \mathcal{L}(V)$ *is the* interface*, and* $n\colon E \to \mathcal{L}(V)$ *and* $l\colon E \to \Sigma$ *are arity-preserving functions respectively giving the* neighbours *and the* label *of each edge.*

The *elements* of a graph are its vertices and edges. The vertices appearing in the interface of a graph are its *sources*. Vertices (resp. elements) which are not sources are called *inner vertices (resp. elements)*. A vertex is *isolated* if it is not in the neighbourhood of any edge. The *size* of a graph G, written $|G|$, is its number of elements. The *arity* of a graph is that of its interface. Two graphs G, H are *isomorphic*, written $G \simeq H$, if their vertex and edge sets are related by structure-preserving bijections (cf. [10, Appendix A]). A graph is:

- *empty* if its only elements are its sources;
- *atomic* if its only inner element is an edge, whose neighbours comprise all the sources.

Our choice of considering ordered hypergraphs with interfaces comes from the need to be able to substitute graphs for edges. A *substitution* is an arity-preserving function from the alphabet to graphs. Given a graph G and a substitution σ, we write $G\sigma$ for the graph

obtained from G by replacing all its a-labelled edges by copies of the graph $\sigma(a)$, identifying pairwise the neighbours of the edges with the interfaces of the copies. We say that G *has shape* H when $G \simeq H\sigma$ for some substitution σ. For instance, the first graph of Figure 1 has the following shapes (amongst other ones):



(Note that the edge orientation is irrelevant in a graph used as a shape, as well as the labelling function – as long as it is injective, which will always be the case in the present paper.)

▶ **Proposition 2.2.** *For all graphs G, H and all substitutions σ, ρ, if $G \simeq H$ and $\sigma(a) \simeq \rho(a)$ for all letters a, then $G\sigma \simeq H\rho$.*

A graph in the sense of Definition 2.1 can be seen as a simple graph, its *skeleton*, by turning all its edges and its interface into cliques (removing duplicate edges if necessary). This operation makes it possible to use the standard notion of treewidth [8, Section 12.4]: the *treewidth* of a graph is that of its skeleton. Equivalently, we can define it as follows.

▶ **Definition 2.3.** *A* tree decomposition *of a graph is a tree whose nodes are labelled by sets of vertices, called* bags*, such that*

- *there is a bag containing all sources,*
- *for each edge there is a bag containing its neighbours,*
- *for each vertex, the bags containing it form a subtree.*

The width *of a tree decomposition is the size of its largest bag minus one. The* treewidth *of a graph is the minimal width of its possible tree decompositions.*

Observe that since the sources of a graph must be contained in a bag for all tree decompositions, the treewidth of a graph is at least its arity minus one. Therefore, at treewidth at most three, we only have graphs of arity up to four. The same constraint holds for the edge arities.

▶ **Proposition 2.4.** *If a graph G and the graphs in the image of a substitution σ all have treewidth at most k, then so does $G\sigma$.*

3 Graph operations

We can now define the operations discussed in the introduction.

▶ **Definition 3.1.** *Let G, H be graphs of arity k.*

- *The* parallel composition *of G and H, $G \parallel H$, is the graph of arity k obtained from the disjoint union of G and H by pairwise merging their sources.*
- *If $k \geq 1$, the* forget *of G, $\mathrm{f}G$, is the graph of arity $k-1$ obtained from G by removing the last vertex from its interface (keeping it in the vertex set).*
- *The* lift *of G, $\mathrm{l}G$, is the graph of arity $k+1$ obtained from G by adding a new isolated vertex, and appending it to its interface.*
- *For a permutation p of $[1, k]$, we denote by pG the graph obtained from G by permuting its interface according to p.*
- *We write $\emptyset_k$ for the empty graph of arity k, omitting k when it is clear from the context.*
- *For a letter $a \in \Sigma$, we also write a for the atomic graph whose edge is labelled a and has its neighbours appearing in the same order as in the interface.*

These operations all arise as substitutions of well chosen small graphs. By Proposition 2.4, this entails that they preserve treewidth (except of course for lifts, which may increase it when reaching the maximal arity). It remains to establish a few structural properties, in order to show that every graph of treewidth at most k can be constructed from the above operations up to arity $k+1$ (Proposition 4.1 below).

A *path* in a graph is a sequence of elements such that any two consecutive elements consist of an edge and one of its neighbours. An *inner path* is a path made only of inner elements, except possibly for the endpoints.

▶ **Definition 3.2.** *A graph is* prime *if it is not empty and all its inner elements are connected by inner paths; it is* full *if none of its sources are isolated.*

Observe that a graph is full prime iff all its elements are connected by inner paths. Amongst the graphs in Figure 1, the first one is full but not prime, the three other ones are prime, and only the last one is full prime.

▶ **Lemma 3.3.** *Every graph is isomorphic to a permutation of lifts of a full graph. This decomposition is unique up to permutation and isomorphism.*

▶ **Lemma 3.4.** *Every graph is isomorphic to a parallel composition of prime graphs. This decomposition is unique up to reindexing and isomorphism.*

We call *(prime) components of a graph* the prime graphs occurring in the latter decomposition. We call *reduced components of a graph* the full prime graphs obtained by removing isolated sources from its components.

We now give two key properties of full prime graphs of bounded treewidth. First, they are always atomic at maximal arity.

▶ **Proposition 3.5.** *Full prime graphs of treewidth at most k and arity $k+1$ are atomic.*

In the case of treewidth at most three, this means that when we decompose a graph of arity four into prime components, then except for a number of four-edges between the sources, we only get non-full components: drawing the graph as a tetrahedron between its sources, the non-atomic components are glued through the faces (or edges, or vertices, or nothing), and the interior of the tetrahedron remains empty. This corresponds to the characterisations of treewidth at most k graphs as partial k-trees [2] or as subgraphs of chordal graphs whose cliques are of size at most $k+1$ [13].

Second, non-atomic full prime graphs have *forget points*, which we define as follows.

▶ **Definition 3.6.** *When x is an inner vertex of a graph G, we write (G, x) for the graph obtained from G by appending x to its interface; if this graph has treewidth at most k then we say that x is a* k-forget point *of G.*

▶ **Proposition 3.7.** *Non-atomic full prime graphs of treewidth at most k have k-forget points.*

4 Terms and axioms

We finally provide a notion of *term* for denoting graphs. We use a multisorted syntax: the family $(\mathcal{T}_k)_{k\in\mathbb{N}}$ of sets of terms of a given arity is defined inductively by the following rules (where p ranges over permutations of $[1,k]$ and a over letters of arity k):

$$\frac{t,u\in\mathcal{T}_k}{t\parallel u\in\mathcal{T}_k}\qquad\frac{t\in\mathcal{T}_k}{\mathrm{l}t\in\mathcal{T}_{k+1}}\qquad\frac{t\in\mathcal{T}_{k+1}}{\mathrm{f}t\in\mathcal{T}_k}\qquad\frac{t\in\mathcal{T}_k}{pt\in\mathcal{T}_k}\qquad\frac{}{\emptyset\in\mathcal{T}_k}\qquad\frac{}{a\in\mathcal{T}_k}$$

This syntax matches the operations in Definition 3.1. Accordingly, we obtain a recursive arity-preserving function g from terms to graphs. We say that a term t *denotes* the graph G, or that t is a *parsing* of G, when $g(t) \simeq G$. For instance, the expressions below the graphs in Figure 1, seen as terms, are parsings of these graphs. Note that the number of inner vertices of a graph is the number of forgets appearing in any of its parsings.

We shall sometimes mention terms and refer implicitly to their graphs; for instance writing that a term is prime to mean that its graph is so.

The *width* of a term is the maximal arity of its subterms, minus one.

▶ **Proposition 4.1.** *A graph has treewidth at most k iff it has a parsing of width at most k.*

Proof. The backward implication is proven by induction on terms, using Proposition 2.4 and the observation that all operations arise as substitutions. For the direct implication, we proceed by lexicographic induction on the size of the graph followed by $k+1$ minus its arity (recall that the arity is always lower or equal to the treewidth plus one).

If the graph has isolated sources, we use Lemma 3.3 to write it as a permutation of lifts and proceed recursively. Otherwise it is full, and we decompose it into primes via Lemma 3.4:

- if there are no components then the graph is empty and has a trivial parsing;
- if there are at least two components then they are smaller, and we proceed recursively;
- otherwise the graph is (full) prime; either it is atomic and it admits a permutation of a letter as a parsing, or, by Proposition 3.7, it can be written as the forget of a graph of the same size but increased arity, which we may parse recursively. ◀

The previous proposition holds for all bounds k on the treewidth. Some of our results below also hold generically, and we will discuss these generalisations in Section 9. Still, from this point on we focus on treewidth at most three.

▶ **Convention 4.2.** *In the remainder, we only work with graphs of treewidth at most three. We simply call* terms *the terms of width at most three, and* forget points *the* 3*-forget points.*

A *(term) substitution* is an arity-preserving function from the alphabet to terms. Such a function σ extends uniquely to a homomorphism $\hat{\sigma}$ from terms to terms:

$$\hat{\sigma}(t \parallel u) \triangleq \hat{\sigma}(t) \parallel \hat{\sigma}(u) \qquad \hat{\sigma}(\mathrm{l}t) \triangleq \mathrm{l}\hat{\sigma}(t)$$
$$\hat{\sigma}(\emptyset) \triangleq \emptyset \qquad \hat{\sigma}(\mathrm{f}t) \triangleq \mathrm{f}\hat{\sigma}(t)$$
$$\hat{\sigma}(a) \triangleq \sigma(a) \qquad \hat{\sigma}(pt) \triangleq p\hat{\sigma}(t)$$

A *context of arity* $i \to o$ is a term of arity o with a single occurrence of a designated letter h of arity i, called the *hole*. Given such a context c and a term t of arity i, we write $c[t]$ for the term c where the hole is replaced by t. Note that $c[t] = \hat{\sigma}_t(c)$ for the substitution σ_t mapping h to t and fixing all other letters.

An *equational theory* is an equivalence relation R on terms, relating only terms of the same arity, and which is closed under contexts and substitutions:

- $(t,u) \in R$ entails $(c[t], c[u]) \in R$ for all contexts c of appropriate arity, and
- $(t,u) \in R$ entails $(\hat{\sigma}(t), \hat{\sigma}(u)) \in R$ for all substitutions σ.

Given two terms t, u, we write $t \cong u$ when $g(t) \simeq g(u)$. Thanks to Proposition 2.2, this relation ($\cong$) is an equational theory, and our goal is to provide a finite list of axioms that generates it.

We give such a list below; most axioms can be written explicitly, but three of them relate rather large terms, which are best presented by their graphs. This is why we rely on the following concept of forget axiom.

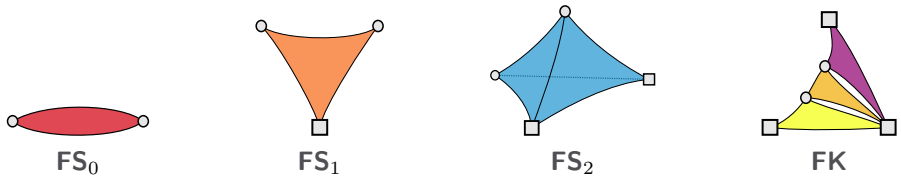


Figure 3 Forget axioms for anchors (swap and kite); x, y are the inner vertices.

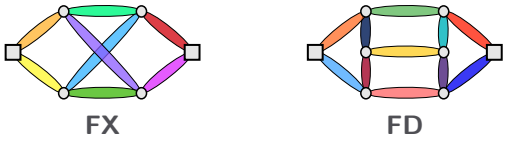


Figure 4 Forget axioms for hard graphs (cross and domino); x, y are the topmost inner vertices.

A *forget axiom* for a graph G with two forget points x, y is an equation of the shape $\mathrm{f}t \equiv \mathrm{f}u$ for some parsings t of (G, x) and u of (G, y).

▶ **Definition 4.3** (Finite axiomatisation). *We write $\equiv$ for the least equational theory containing the following axioms, for letters a, b, c of appropriate arities:*
A1. $a \parallel (b \parallel c) \equiv (a \parallel b) \parallel c$, $a \parallel b \equiv b \parallel a$, *and* $a \parallel \emptyset \equiv a$;
A2. $pqa \equiv (p \circ q)a$ *for all permutations* p, q, *and* $\mathrm{id}a \equiv a$ *where* id *is the identity permutation;*
A3. $p(a \parallel b) \equiv pa \parallel pb$ *and* $p\emptyset \equiv \emptyset$ *for all permutations* p;
A4. $\mathrm{l}(a \parallel b) \equiv \mathrm{l}a \parallel \mathrm{l}b$ *and* $\mathrm{l}\emptyset \equiv \emptyset$;
A5. $p\mathrm{f}a \equiv \mathrm{f}\dot{p}a$ *and* $\mathrm{l}pa \equiv \dot{p}\mathrm{l}a$ *where* $\dot{p}$ *is the extension of a permutation* p *of* $[1, k]$ *to* $[1, k+1]$;
A6. $\mathrm{l}\mathrm{f}a \equiv \mathrm{f}r\mathrm{l}a$ *and* $\mathrm{l}\mathrm{l}a \equiv r\mathrm{l}\mathrm{l}a$ *for the permutation* r *that swaps the last two elements;*
A7. $\mathrm{f}a \parallel b \equiv \mathrm{f}(a \parallel \mathrm{l}b)$;
FS. $\mathrm{f}\mathrm{f}a \equiv \mathrm{f}\mathrm{f}ra$ *for the permutation* r *that swaps the last two elements;*
as well as three forget axioms for the graphs with forget points **FK**, **FX** *and* **FD** *in Figures 3&4 (for some arbitrary choice of interface ordering, edge orientation, and injective edge labelling).*

The first eight items are universally quantified over all appropriate arities. For instance, associativity of parallel composition (in **A1**) is an axiom at each arity, and a, b and $\emptyset$ may have arity up to three in **A4**. Since we restrict globally to arities up to four, the list of axioms is nevertheless finite.

Also note that even though the axioms are expressed using letters, they yield laws which hold under all term substitutions since $\equiv$ is defined as an equational theory.

FS comprises three axioms depending on the arity of a (2, 3, or 4), which we shall sometimes refer to as $\mathbf{FS}_i$ (with $i = 0, 1$, or 2, respectively). These are forget axioms for the first three graphs in Figure 3, and we could have presented them as such. We cannot express $\mathbf{FS}_3$ as it would require an edge of arity five; **FK** intuitively is a weakened version of it, expressible at treewidth three.

We shall see at the end of Section 5 that the concrete choice of parsings for the forget axioms **FK**, **FX**, and **FD** is irrelevant thanks to **A1-7**. The interested reader may find concrete equations for them in [10, Appendix C].

▶ **Proposition 4.4** (Soundness). *If $t \equiv u$ then $t \cong u$.*

Proof. Since $\cong$ is an equational theory, it suffices to check that the two members of each axiom denote the same graph. This is by definition for the forget axioms, and a routine verification for the other ones. ◀

The axioms given textually in Definition 4.3 (**A1-7** and **FS**) intuitively correspond to those given for arbitrary graphs by Courcelle and Engelfriet [7, p.117], restricted to terms of width at most three. (Modulo some translation since our syntaxes and sets of operations differ.) This finite restriction is however incomplete for graphs of treewidth at most three: in their completeness proof, Courcelle and Engelfriet need axioms of the form $\mathbf{FS}_i$ for arbitrary large arity i, even when dealing with graphs of small treewidth. Somehow, we prove below that the forget axioms $\mathbf{FS}_0$, $\mathbf{FS}_1$, $\mathbf{FS}_2$, **FK**, **FX**, and **FD** suffice at treewidth at most three to fulfil the role played by this infinite family of axioms. Also observe that while **FK** easily generalises to deal with an arbitrary treewidth k (by replacing the source touching the three edges by $k-2$ sources), this is not the case for **FX** and **FD**.

5 Outline of the completeness proof

Using the first seven axioms, we easily obtain the following proposition. Together with Lemmas 3.3 and 3.4, it makes it possible to normalise terms and to focus on full prime ones.

▶ **Proposition 5.1.** *For all terms t, letters a, and graphs G, H, we have:*
1. *if $g(t) \simeq \emptyset$ then $t \equiv \emptyset$;*
2. *if $g(t) \simeq a$ then $t \equiv a$;*
3. *if $g(t) \simeq pG$ then there is a parsing u of G such that $t \equiv pu$;*
4. *if $g(t) \simeq \mathrm{l}G$ then there is a parsing u of G such that $t \equiv \mathrm{l}u$;*
5. *if $g(t) \simeq G \parallel H$ then there are parsings u of G and v of H such that $t \equiv u \parallel v$.*

Proof sketch. All items but the third are proven by induction on t. The first three items only need **A1-4**; the fourth one needs **A5-6**; the last one rests on the fourth one and **A7**. ◀

An item is patently missing from the previous proposition, for the forget operation. In fact, a statement similar to the third and fourth items does not hold for the forget operation: when t is a parsing of $\mathrm{f}G$, nothing guarantees that G has a parsing at all: its treewidth may be four; we need to restrict to those cases where the forgotten vertex is a forget point.

Given a term t denoting a graph G with an inner vertex x, we say that t *reaches* x if there is a parsing t' of (G, x) such that $t \equiv \mathrm{f}t'$. We can easily prove the following property.

▶ **Lemma 5.2.** *Every non-atomic full prime term reaches some forget point.*

Proof. By induction on t, using Proposition 5.1 until we find a forget operation. ◀

However, this property is not enough, because it gives no control on the forget point which is reached; instead, we would like to obtain:

▶ **Statement 5.3** (Reaching forget points)**.** *Full prime terms reach all their forget points.*

This statement holds, but we do not know how to prove it directly: we only get it *a posteriori*, from the completeness. We prove a variant of it in the next section (Lemma 7.1), for *anchors*.

Instead, the cornerstone of our proof is the following lemma: any two parsings of a non-atomic full prime graph may reach a common forget point.

▶ **Lemma 5.4** (Forget point agreement)**.** *For all parsings t, u of a non-atomic full prime graph, there is a forget point reached by both t and u.*

The rest of the paper is devoted to proving this lemma, with which we conclude as follows.

▶ **Theorem 5.5** (Completeness)**.** *For all parsings t, u of a given graph, we have $t \equiv u$.*

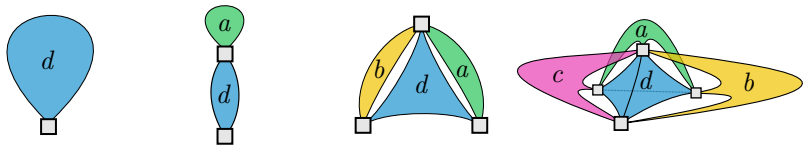


Figure 5 The series operations $\mathrm{s}(;d)$, $\mathrm{s}(a;d)$, $\mathrm{s}(a,b;d)$, and $\mathrm{s}(a,b,c;d)$.

Proof. We follow the same pattern as in the proof of the forward implication of Proposition 4.1: we proceed by lexicographic induction on the size of the graph followed by 4 minus its arity. If the graph has isolated sources, we use Lemma 3.3 and Proposition 5.1(3,4) to rewrite the parsings into permuted lifts and proceed recursively. Otherwise it is full, and we decompose it into primes via Lemma 3.4:

- if there are no components then both parsings are equivalent to $\emptyset$ by Proposition 5.1(1);
- if there are at least two components then we use Proposition 5.1(5) on both parsings and we proceed recursively.
- otherwise the graph must be (full) prime. Either it is atomic and we conclude by Proposition 5.1(2,3), or Lemma 5.4 gives us two terms t', u' such that $t \equiv \mathrm{f}(t')$, $t' \cong u'$, and $\mathrm{f}(u') \equiv u$, and we can conclude since $t' \equiv u'$ follows by induction hypothesis. ◀

We have only used Axioms **A1**-**7** up to this point. We prove below that those axioms are complete when there are few forget points. Say that a graph is *easy* if each of its non-atomic reduced components G has only one forget point x, and in turn (G, x) is easy. (This definition is well-founded by the same lexicographic ordering as we used in the above proof.)

▶ **Proposition 5.6.** *For all parsings t, u of an easy graph, we have $t \equiv u$.*

Proof. By adapting the previous completeness proof. Lemma 5.2 may replace Lemma 5.4 when we attain non-atomic full prime graphs: those are reduced components of the starting graph and since those have only one forget point, all their parsings reach that one. ◀

None of the graphs in Figures 3 and 4 are easy: they are all full prime, each with two forget points x, y (plus two symmetrical ones for **FX** and **FD**). Nevertheless, adding either x or y to their interface makes them easy (cf. [10, Lemma D.1]). This is why the precise choice of parsings does not matter when we use them as forget axioms in Definition 4.3.

6 Series decompositions

As explained in the introduction, there is a *series* operation on graphs which plays an important role. We slightly generalise it here, and we show how to use it to analyse graphs with a forget point.

Given k graphs $G_1, \dots, G_k$ of arity k and a graph H of arity $k+1$, we define the following graph of arity $k+1$, where p_i denotes the permutation which swaps i and $k+1$.

$$\mathrm{s}(G_1, \dots, G_k; H) \triangleq p_1 \mathrm{l} G_1 \parallel \cdots \parallel p_k \mathrm{l} G_k \parallel H$$

As is explicit from its definition, this operation on graphs is also a derived operation on terms. We illustrate its behaviour at each arity in Figure 5. We recover the operation from the introduction when the last argument is empty, and using a forget operation: we have $\circ(u, v, w) = \mathrm{fs}(u, v, w; \emptyset)$.

We use this operation in order to decompose full prime graphs along a given inner vertex.

▶ **Proposition 6.1** (Series decomposition). *For all full prime graphs G of arity k with an inner vertex x, there are graphs $G_1, \dots, G_k, H$ such that $(G, x) \simeq \mathrm{s}(G_1, \dots, G_k; H)$ and*
1. *all components of H are full, and*
2. *if the jth source of a component of G_i is isolated, then $i < j$.*

Such a decomposition is unique up to isomorphism.

We call G_i the *ith series argument*, and H the *series factor* of such a decomposition (at x).

Proof. We decompose (G, x) into prime components, which we classify according to their isolated sources. Since G is full prime, x is never isolated. Full components go into the series factor. Components where the first source is isolated go into the first series argument (under the permutation p_1, the first source gets swapped with x). Components where the second source is isolated but not the first one go into the second series argument. *Et caetera.* ◀

Also note that we can follow such decompositions at the term level, modulo **A1**-**7**:

▶ **Proposition 6.2.** *If a term t reaches the last source of a graph of the form $\mathrm{s}(G_1, \dots, G_k; H)$, then there are parsings $u_1, \dots, u_k, v$ of $G_1, \dots, G_k, H$ such that $t \equiv \mathrm{fs}(u_1, \dots, u_k; v)$.*

Proof. Consequence of Proposition 5.1. ◀

7 Anchors

In this section we define the concept of *anchors* – inner vertices with specific properties, and we prove the following variant of Statement 5.3:

▶ **Lemma 7.1** (Reaching anchors). *Full prime terms reach all their anchors.*

This lemma implies Lemma 5.4 when the considered graph has an anchor, by applying it to both parsings. Therefore, once Lemma 7.1 proved, it will only remain to prove Lemma 5.4 for anchor-free graphs (Section 8).

The definition of anchor is the following; we only consider them in full prime graphs.

▶ **Definition 7.2.** *An* anchor *in a (full prime) graph G is an inner vertex x such that either:*
1. *there are no full prime components in (G, x), or*
2. *there is an edge whose neighbours comprise at least x and all sources of G, or*
3. *there are two or more full prime components in (G, x).*

Using series decompositions, x is an anchor whenever the series factor at x is empty, or has only one edge, or has at least two components.

This last condition generalises the usual notion of cutvertex [8, Section 1.4]: such anchors at arity zero when there are only binary edges are exactly cutvertices as usual. Removing them disconnects the graph. The second branch in our definition may only arise with sources and/or hyperedges; it is convenient to have it for the present proof, but it would probably be more natural to disallow it in other contexts. Thanks to this alternative, all inner vertices in the graphs in Figure 3 are anchors. In contrast, the graphs in Figure 4 have no anchors.

We prove below that all forget points are anchors at arity three. The converse holds at all arities, but we only get it *a posteriori*, as a consequence of Lemma 7.1.

▶ **Proposition 7.3.** *At arity three, all forget points are anchors.*

Proof. If x is a forget point of a graph G of arity three, the full prime components of (G, x) must be atomic by Proposition 3.5. Thus x is an anchor, by either the first or the second condition in Definition 7.2. ◀

We also find anchors easily at arities zero and one.

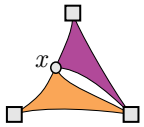
▶ **Proposition 7.4.** *Non-atomic full prime graphs of arity in* $\{0, 1, 3\}$ *have some anchor.*

Proof. Every inner vertex is an anchor at arity zero, either because it is isolated (anchor of the first kind), or because it belongs to some edge (anchor of the second kind). Every immediate neighbour of the only source is an anchor at arity one (of the second kind). At arity three we use Propositions 7.3 and 3.7. ◀

We first prove Lemma 7.1 for graphs of arity three, and then we show how to reduce the other cases to that one (Section 7.2).

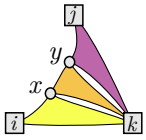
7.1 Parsing on anchors at arity three

A *checkpoint* between two vertices y, z of a graph G is an inner vertex x such that every inner path from y to z goes through x. When y, z are not mentioned explicitly, they are implicitly assumed to be sources. (This definition coincides with that from [5] at arity two.) Analysing the components of (G, x) by the sources they touch, we see that the checkpoints of a graph G of arity three are the inner vertices x for which G has the following shape:



This characterisation shows that checkpoints are anchors of the first kind in Definition 7.2. Another important property at arity three is that if there are two anchors, then they must be checkpoints for the same sources:

▶ **Proposition 7.5.** *At arity three, every full prime graph with two distinct anchors* x, y *has the following shape:*



Proof of Lemma 7.1 at arity three. We proceed by induction on the size of the graph.

Let t be a parsing of a full prime graph G of arity three, with an anchor x. By Lemma 5.2, t reaches some forget point y. If $x = y$ then we are done. Otherwise, y is a second anchor by Proposition 7.3, and G has the shape given by Proposition 7.5. Call A the graph between i,k,y, and B the one between j, k, y. Since t reaches y we have a parsing t' of (G, y) such that $t \equiv \mathrm{f}t'$. By analysing the full prime decomposition of (G, y), we can put t' under the form $(j4)\mathrm{l}u \parallel (i4)\mathrm{l}v$, where u is a parsing of A and v a parsing of B. Then we observe that x is a checkpoint in A, and thus an anchor, so that u reaches x by induction hypothesis. Therefore, $u \equiv \mathrm{f}u'$ for some parsing u' of (A, x). By analysing the full prime decomposition of (A, x) as before, we deduce that t reaches x using **FK**. ◀

7.2 Parsing on anchors at lower arities

We now finish the proof of Lemma 7.1 by showing how to reduce the other cases to that of arity three. We use for that the two following lemmas. The first one intuitively makes it possible to zoom on a specific inner vertex by going under a forget operation. The second one states that under some conditions, an anchor in $\mathrm{f}^i G$ is also an anchor in G, allowing to use the case of arity three.

▶ **Lemma 7.6.** *Let t be a term of arity up to two with an inner vertex x in a full prime component. There is a term u such that $t \equiv \mathrm{f}u$ and x is either the last source of u or an inner vertex of a full prime component of u.*

▶ **Lemma 7.7.** *For $i > 0$ let $\mathrm{f}^i G$ be a full prime graph of arity k and x, y two distinct inner vertices in a full prime component C of G. If x is an anchor of $\mathrm{f}^i G$ and is in one of the first k series arguments of the series decomposition of C at y, then x is a checkpoint in C.*

(Note that the only isolated sources of the components of the i last series arguments in the decomposition of C as above, are the i forgotten sources in $\mathrm{f}^i G$.)

Before we finish the proof of Lemma 7.1, also observe that **FS** implies $\mathrm{f}^j pu \equiv \mathrm{f}^j qu$ for all terms u of arity k and permutations p, q agreeing on the first $k - j$ elements.

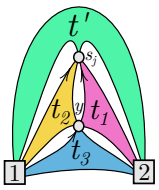
Proof of Lemma 7.1, at all arities. We prove the following generalisation, by induction on the lexicographic product of $|C|$ and $3 - k$:

> For all terms t of arity $k \leq 3$, for all $i \leq k$, if $\mathrm{f}^i t$ is full prime with an anchor x in a full prime component C of t, then $\mathrm{f}^i t$ reaches x.

If $k < 3$ then we use Lemma 7.6 to obtain u such that $\mathrm{f}^i t \equiv \mathrm{f}^{i+1} u$; if x is the last source of u then we are done, otherwise we conclude by induction, since x lies in a component of C.

We now assume $k = 3$, and we rewrite t as $t_C \parallel t'$, where t_C is the full prime component containing x. By Lemma 5.2, t_C reaches some forget point y. Take a series decomposition $t_C \equiv \mathrm{fs}(t_1, t_2, t_3; u)$ of t_C at y. The series factor u cannot contain x since all its components are atomic by Proposition 3.5. Thus x must be in a series argument.

If x is in one of the first $3 - i$ series arguments, then by Lemma 7.7 x must be a checkpoint, and thus an anchor of t_C. Using Lemma 7.1 at arity three (Section 7.1), we obtain that t_C reaches x. So does t by **A7**, and finally $\mathrm{f}^i t$ by **FS**. Otherwise, x is in one of the last i series arguments, say the jth one. The jth source s_j is the only source that is necessarily isolated in this argument. It is a source that is forgotten by the operations f^i (see the figure below for an illustration). Our goal is to "swap" this jth source for y in order to decrease the size of C. Indeed the new component containing x will then be contained in t_3, which is smaller in size than C.



Let S be the set of non-isolated sources in the component C' of $s(t_1, t_2, t_3; u)$ containing x. Using **FS** we get a term t'' such that $\mathrm{f}^i t \equiv \mathrm{f}^{|S|} t''$ and t'' denotes $g(t)$ with the inner vertices in S upgraded as sources. By definition of t'', C' is a full prime component of it. Furthermore, up to isolated sources, C' is the same component as the one of $s(t_1, t_2, t_3; u)$ containing x. This gives $|C'| < |C|$ and we conclude by induction on $\mathrm{f}^{|S|} t''$. ◀

At this point, we have used axioms **A1**-**7**, **FS**, and **FK**, but not **FX** and **FD**.

Figure 6 Separation pairs $(y \diamond x)$, and partial order on vertices $(y \prec x)$.

8 Separation pairs

We call *hard* the non-atomic anchor-free full prime graphs. Examples of such graphs were given in Figure 2. Hard graphs have arity two by Propositions 7.4 and 3.5, and it only remains to prove Lemma 5.4 for these graphs.

A *forget pair* in a graph G is a pair (x, y) of inner vertices such that (G, x, y) has treewidth at most three. A parsing t of a graph G *reaches a pair* (x, y) of inner vertices if there is some parsing t' of (G, x, y) such that $t \equiv \mathrm{fft}'$. By definition, a term may only reach forget pairs. By **FS**, a term reaches (x, y) iff it reaches (y, x); it follows that every term reaching (x, y) reaches both x and y.

Given two inner vertices x, y of a full prime graph of arity two, we write $y \diamond x$ when the graph has the shape on the left of Figure 6, and $y \prec x$ when the graph has the shape on the right (equivalently, when y is a checkpoint between x and some source). In the first case, we say that (x, y) is a *separation pair*. We prove a few properties about such inner vertices.

▶ **Lemma 8.1.** *If $y \prec x$ or $y \diamond x$ in a full prime graph G, then y is an anchor in (G, x), and every parsing of G reaching x also reaches (x, y).*

Proof. In both cases, there is no full prime component in (G, x, y), so that y is indeed an anchor in (G, x). Now if $t \equiv \mathrm{f}t'$ for some parsing t' of (G, x), then t' reaches y by Lemma 7.1: $t' \equiv \mathrm{f}t''$ for some parsing t'' of (G, x, y). Thus t reaches (x, y). ◀

It follows that a full prime term reaches a separation pair iff it reaches any of its constituents.

We write $S(x)$ for the series factor of a graph at some inner vertex x. When the graph is hard, x is not an anchor, so that $S(x)$ must be full prime and cannot contain just one edge.

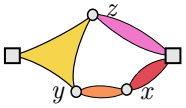
▶ **Proposition 8.2.** *For all hard graphs, $\prec$ is well-founded.*

Proof. Observe that if $y \prec x$ then $|S(y)| < |S(x)|$. ◀

▶ **Proposition 8.3.** *For all vertices x, y, z such that $y \prec x$ and $x \diamond z$, we have $y \diamond z$.*

Proof. Consider an inner path between the two sources; we have to show that it visits either y or z. It visits either x or z since (x, z) is a separation pair. If it visits x then it also visits y since y is a checkpoint between x and one of the two sources. ◀

When the graph is hard in the previous proposition, it must have the following shape:

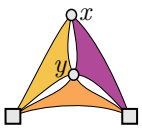


In such a case, we intuitively prefer the separation pair (y, z) to (x, z). This leads us to the notion of minimal separation pair. A vertex x is *minimal* if there is no vertex y such that $y \prec x$. A pair of vertices is minimal when its two elements are so.

The proposition below makes it possible to get separation pairs out of minimal vertices.

▶ **Proposition 8.4.** *Let x be a forget point in a hard graph. For all forget points y of $S(x)$, we have either $y \prec x$ or $y \diamond x$.*

Proof. Let us classify the components of (G, x, y) by their isolated sources. Full components cannot exist as by Proposition 3.5 they would be edges and both x and y would be anchors in G. So G must have the following shape:



If there is no inner path avoiding y between the sources in the bottom component, then $y \diamond x$. If there is no inner path avoiding y from x to one of the sources, then $y \prec x$. Otherwise there are at least two full prime components in (G, y), making y an anchor of G, which contradicts G being hard. ◀

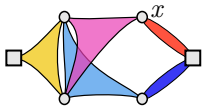
It follows easily that every hard graph has some separation pair – even a minimal one, which will be useful to reduce the number of cases to study. However, we need to be more precise and to keep track of terms.

▶ **Lemma 8.5.** *Every hard term reaches some minimal separation pair.*

Proof. Let t be a hard term. By Lemma 5.2 t reaches some forget point x, which we can choose minimal by Proposition 8.2 and Lemma 8.1. Accordingly, let t' be a parsing of (G, x) such that $t \equiv \mathrm{f}t'$. As $S(x)$ is full prime and non-atomic, it has a forget point y. Since x is minimal, (x, y) is a separation pair by Proposition 8.4. We can choose y to be minimal by Proposition 8.3, and t reaches (x, y) by Lemma 8.1. ◀

As before with forget points, separation pairs are not unique (even minimal ones), and we need to show how to move from one separation pair to another.

When studying the possible shapes of a hard graph H with distinct separation forget pairs, we end up with a few shapes S such as the following one:



In such situations, we know that (H, x) has treewidth at most three, but (S, x) has treewidth four. Therefore, for an appropriate notion of *minor*, (S, x) cannot be a minor of (H, x): the treewidth may not increase when taking minors. We use this information in order to refine the actual shape of H.

We use *sourced simple graphs* in order to define such a notion of minor. Those are triples (V, E, S) made of a set V of *vertices*, a set E of unordered binary *edges*, and a subset $S \subseteq V$ of *sources*. We write K_k for the clique over k sources. *Tree decompositions* and *treewidth* [8, Section 12.4] are adapted to sourced simple graphs by requiring the subset of sources to be contained in some bag. A *sourced minor of a (sourced simple) graph* is a (sourced simple) graph obtained from it by a sequence of the following operations: remove an edge, contract an edge, remove an isolated vertex. Those operations do not increase the treewidth.

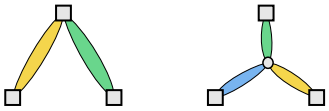
The *footprint* of a graph is the sourced simple graph obtained from it by replacing each edge by a clique over its neighbours, forgetting labels, and turning the interface into a mere set – note that we do not add a clique on the sources. A graph and its footprint have the same tree decompositions, and thus the same treewidth. A *sourced minor of a graph* is a sourced minor of its footprint.

Note that graphs excluding K_3 as a sourced minor need not be acyclic, as cycles may occur away from the sources. The point of using sourced minors is that they behave well with respect to substitutions, and thus shapes:

▶ **Proposition 8.6.** *Given a graph G and a substitution σ, if K_k is a sourced minor of $\sigma(a)$ for all letters a of arity k, then the footprint of G is a sourced minor of $G\sigma$.*

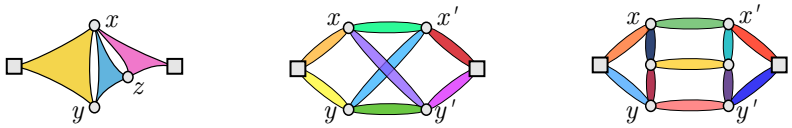
Now observe that full prime graphs of arity two admit the sourced edge K_2 as a sourced minor. At arity three, we have the following property instead.

▶ **Proposition 8.7.** *Every graph of arity three either has the sourced triangle K_3 as a sourced minor, or has one of the two following shapes:*



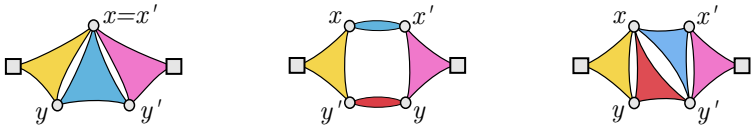
Let us continue our example and come back to the previous hard graph H. Assume that the binary edges in the given shape are instantiated by full prime graphs, and consider the three graphs instantiating the ternary edges. If those three graphs have the triangle as a sourced minor, then the footprint of (S, x) must be a sourced minor of (H, x), which is not possible. Therefore, at least one of them must have one of the shapes given by Proposition 8.7. By considering the different possibilities and continuing the same kind of reasoning, we end up proving that in this case, H must have shape **FX**. In the general case, we obtain the following classification.

▶ **Proposition 8.8.** *Let G be a hard graph with two minimal separation forget pairs (x, y) and (x', y'). Possibly up to swapping x and y, and/or x' and y', either (x, y') is a separation pair (**Case Z**), or G has one of the following shapes, where in the leftmost case, z is minimal and x', y' appear in the rightmost component:*



Case IH **Case X** **Case D**

The proof we provide in [10, Appendix F] actually establishes that in the case where (x, y') is a separation pair, then G has one of the following shapes:



Case Z1 **Case Z2** **Case Z3**

We finally deduce that hard terms reach all their minimal separation forget pairs.

▶ **Lemma 8.9.** *Let G be a hard graph with minimal separation forget pair (x', y'). All parsings t of G reaching a minimal separation pair (x, y) also reach (x', y').*

Proof. We fix G, x', y', we proceed by induction on the size of the component of (G, x, y) containing x', and we use Proposition 8.8. If (x, y') is a separation pair (**Case Z**), then we use Proposition 8.1 twice: t reaches x, thus (x, y'), thus y' in particular, thus (x', y'). In

Case IH we proceed by induction on (x, z), which is reached by t by similar reasoning, and for which the induction measure decreases. In **Case X** and **Case D**, we conclude directly by **FX** and **FD**, respectively. (In those latter two cases, we first need to rewrite t so that it agrees syntactically with the shape of the axiom we use. We use Lemma 7.1 for that, which requires showing that some of the inner vertices are anchors in appropriate subgraphs. To do so, we exploit the fact that G is hard, so that the subgraphs corresponding to the edges of the exhibited shape must be connected enough.) ◀

Together with Lemma 8.5, Lemma 5.4 follows, which concludes our completeness proof.

9 Conclusion and future work

We have provided a finite presentation of graphs of treewidth at most three. Our axiomatisation comprises axioms for dealing with full prime decompositions (**A1-7**), axioms for reaching anchors (**FS**, **FK**), and two axioms for hard graphs (**FX** and **FD**).

An obvious question for future work is whether the approach presented here generalises to the case of graphs of treewidth at most k.

It does so up to Section 7. The syntax we use readily characterises those graphs (Proposition 4.1), and the seven first axioms as well as the results in Section 5 are not specific to the case $k = 3$. Axioms for anchors (Figure 3) also generalise. Accordingly, our results about series decomposition and anchors (Lemma 7.1) extend easily. In particular, non-atomic full prime graphs of arity $0, 1$ and k would always have an anchor.

This means we also have finite presentations for bounds $k = 1, 2$. (For $k = 2$ this axiomatisation differs from the one previously proposed [5, 9], because the chosen syntax is different: there, graphs of arity three are not considered, and parallel composition may be applied to graphs of distinct arities.)

However, the results from Section 8 are specific to the case $k = 3$, and getting finite presentations for larger values of k seems difficult. Following the strategy presented here, we would need to prove the forget point agreement lemma (Lemma 5.4) for new hard graphs, which may have arities between 2 and $k - 1$.

Still, we would like to emphasise the utility of such a generalisation in the context of graph theory. Robertson and Seymour proved [14] that for all integers k there must be a finite list of excluded minors characterising the class of treewidth at most k graphs. Consider a minimal one amongst them, say H, along with a maximal clique over vertices $x_1, \dots, x_i$. It is not difficult to prove that either H is K_{k+2}, or $(H, x_1, \dots, x_i)$ is hard. Thus, studying hard graphs of treewidth $k + 1$ might lead to a better understanding of excluded minors for treewidth at most k.

References

1 Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *J. ACM*, 40(5):1134–1164, 1993. doi:10.1145/174147.169807.

2 Stefan Arnborg and Andrzej Proskurowski. Characterization and recognition of partial 3-trees. *SIAM Journal on Algebraic Discrete Methods*, 7(2):305–314, 1986. doi:10.1137/0607033.

3 Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973. doi:10.1016/0097-3165(73)90016-2.

4 Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Prívara and Peter Ruzicka, editors, *Proc. MFCS*, volume 1295 of *LNCS*, pages 19–36. Springer, 1997. doi:10.1007/BFB0029946.

5 Enric Cosme-Llópez and Damien Pous. K4-free graphs as a free algebra. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *Proc. MFCS*, volume 83 of *LIPIcs*, pages 76:1–76:14. Schloss Dagstuhl, 2017. doi:10.4230/LIPICS.MFCS.2017.76.

6 Bruno Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In Neil Robertson and Paul D. Seymour, editors, *Proc. Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1991.

7 Bruno Courcelle and Joost 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. URL: http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR.

8 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

9 Christian Doczkal and Damien Pous. Treewidth-two graphs as a free algebra. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *Proc. MFCS*, volume 117 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl, 2018. doi:10.4230/LIPICS.MFCS.2018.60.

10 Amina Doumane, Samuel Humeau, and Damien Pous. A finite presentation of graphs of treewidth at most three, 2024. Version of this paper with the appendix. URL: https://hal.science/hal-04560570.

11 Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8:171–186, 1976. doi:10.1007/BF01917434.

12 Damien Pous. Hypergraphs. Software, swhId: swh:1:dir:028863b2f75dde258591611a6c7c165e289db890 (visited on 2024-06-17). URL: https://perso.ens-lyon.fr/damien.pous/hypergraph/.

13 Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

14 Neil Robertson and Paul D. Seymour. Graph minors. XX. wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. doi:10.1016/J.JCTB.2004.08.001.

15 Daniel P. Sanders. *Linear algorithms for graphs of tree-width at most four*. PhD thesis, Georgia Institute of Technology, 1993. URL: http://hdl.handle.net/1853/30061.

16 Petra Scheffler. Linear-time algorithms for NP-complete problems restricted to partial k-trees. Technical report, Academy of Sciences of the GDR, 1987.

17 Thomas V. Wimer. *Linear Algorithms on k-Terminal Graphs*. PhD thesis, Clemson University, 1993. URL: https://tigerprints.clemson.edu/arv_dissertations/28.