K_4-Free Graphs as a Free Algebra*

Enric Cosme-Llópez[1] and Damien Pous[2]

1 **Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, Lyon, France**
2 **Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, Lyon, France**

Abstract

Graphs of treewidth at most two are the ones excluding the clique with four vertices as a minor. Equivalently, they are the graphs whose biconnected components are series-parallel.

We turn those graphs into a free algebra, answering positively a question by Courcelle and Engelfriet, in the case of treewidth two. First we propose a syntax for denoting them: in addition to series and parallel compositions, it suffices to consider the neutral elements of those operations and a unary transpose operation. Then we give a finite equational presentation and we prove it complete: two terms from the syntax are congruent if and only if they denote the same graph.

1998 ACM Subject Classification G.2.2 Graph Theory, F.4.3 Formal Languages.

Keywords and phrases Universal Algebra, Graph theory, Axiomatisation, Graph minors.

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.76

1 Introduction

The notion of treewidth is a cornerstone in (algorithmic) graph theory [16]. It measures how close a graph is to a forest, and classes of graphs of bounded treewidth often enjoy good computational properties. For instance, graph homomorphism (and thus k-colouring) becomes polynomial-time [19, 7, 21], so does model-checking of Monadic Second Order (MSO) formulae, and satisfiability of MSO formulae becomes decidable, even linear [9]. (See the monograph of Courcelle and Engelfriet about monadic second order logic on graphs [13].)

Here we focus on graphs of treewidth at most two. They coincide with the *partial* 2*-trees*, with the K_4*-free* graphs (those that exclude the clique with four vertices (K_4) as a *minor*), and with the graphs whose biconnected components are *series-parallel* [18, 5].

We consider the set Gph of directed graphs with edges labelled with letters $a, b, \ldots$ in some alphabet Σ, and with two distinguished vertices, called the *input* and the *output*. We represent such graphs as usual, using an unlabelled ingoing (resp. outgoing) arrow to denote the input (resp. output). Such graphs can be composed:

- in *parallel* by putting them side by side, merging their inputs, and merging their outputs;
- in *series* by putting them one after the other and merging the output of the first one with the input of the second one.

Every letter of the alphabet gives rise to a graph consisting of two vertices (the input and the output), and a single edge from the input to the output, labelled with that letter.

If we allow only those operations, we obtain the series-parallel graphs, and it is easy to see that these form the free algebra over the signature $\langle \parallel, \cdot \rangle$, where $\parallel$ is an associative-commutative binary operation (parallel composition), and $\cdot$ is an associative binary operation

* An extended version of this abstract, including proofs, is available on HAL [8]. This work was supported by the European Research Council (ERC) under the Horizon 2020 programme (CoVeCe, grant agreement No 678157) and the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007)

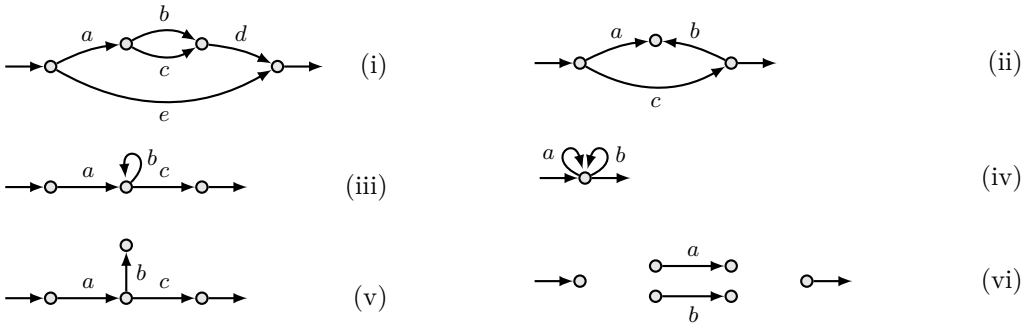


Figure 1 Some graphs of treewidth at most two

(series composition). For instance, the terms $((a{\cdot}(b \parallel c)){\cdot}d) \parallel e$ and $e \parallel (a{\cdot}((c \parallel b){\cdot}d))$ both denote the graph (i) in Figure 1; they are equal up to associativity of $\cdot$ and commutativity of $\parallel$. Parallel composition is not idempotent: $(a{\cdot}b) \parallel (a{\cdot}b)$ and $a{\cdot}b$ denote distinct graphs.

However, we cannot denote all graphs of treewidth at most two in such a way.

First the notion of treewidth does not depend on the orientation of the edges. For instance, the graph (ii) has treewidth two, yet it is not the image of a term in the previous syntax. To this end, we add a unary operation $\cdot^\circ$ to our signature, which we interpret in graphs as the exchange of input and output. Doing so, the terms $(a{\cdot}b^\circ) \parallel c$ and $(b{\cdot}a^\circ)^\circ \parallel c$ denote the graph (ii), and series-parallel graphs with converse become a free algebra when we ask that $\cdot^\circ$ is an involution that distributes over $\parallel$ and satisfies $(a{\cdot}b)^\circ = b^\circ{\cdot}a^\circ$.

Second, the treewidth of a graph does not depend on self-loops, so that the graph (iii) actually has treewidth one (it is a tree once we remove the self-loop). There it suffices to add a constant, 1, interpreted as the graph with a single vertex (both input and output), and no edge. Doing so, the graph (iii) is denoted by $a{\cdot}(1 \parallel b){\cdot}c$. While the constant 1 is clearly a neutral element for series composition $(\cdot)$, axiomatising its interactions with parallel composition is much harder, and is actually one of the key contributions of the present work. For instance, the equation $(1 \parallel a){\cdot}(1 \parallel b) = 1 \parallel a \parallel b$ belongs to the theory as both sides denote the graph (iv).

Up-to this point, we have recovered the syntax of *allegories* [20], and the graphs associated to the terms are precisely the ones Freyd and Scedrov use to obtain that the theory of representable allegories is decidable, yet not finitely presentable [20, p. 210].

But we still miss some graphs, like (v). One can also remark that we only obtain connected graphs using the above operations. Instead, treewidth allows disconnected graphs: a graph has a given treedwidth if and only if all of its connected components do. Surprisingly, it suffices to add a second constant, $\top$, interpreted as the disconnected graph with no edges and two distinct vertices (the input and the output). This allows us to obtain disconnected graphs, but also to get a term for the connected graph (v), namely, $a{\cdot}((b{\cdot}\top) \parallel c)$. Again while it is clear that this constant is a neutral element for parallel composition ($\parallel$), capturing its interactions with the other operations is non-trivial. For instance, $\top{\cdot}a{\cdot}\top{\cdot}b{\cdot}\top$ and $\top{\cdot}b{\cdot}\top{\cdot}a{\cdot}\top$ both denote the graph (vi), and should thus be equated.

To sum up, the set Gph of graphs forms an algebra for the signature $\langle \cdot_2, \parallel_2, \cdot^\circ_1, 1_0, \top_0 \rangle$; the various operations of this algebra are depicted in Figure 2.

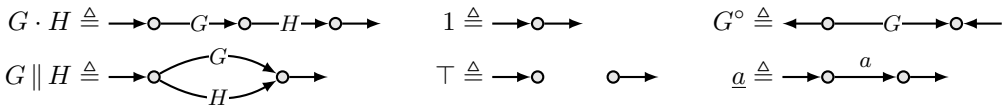
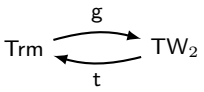


Figure 2 The 2p-algebra of graphs and the graph of a letter

Write Trm for the set of terms over the alphabet Σ and TW_2 for the set of graphs of treewidth at most two when an extra edge is added between input and output[1]. One easily proves that the latter set actually forms a subalgebra of the algebra of graphs. Therefore, the function interpreting each term as a graph actually gives a function $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$.

We first prove that this function has a right-inverse: we define a function $\mathsf{t} : \mathsf{TW}_2 \to \mathsf{Trm}$ such that for all graph $G \in \mathsf{TW}_2$, $\mathsf{g}(\mathsf{t}(G))$ is isomorphic to G:



$$\mathsf{g}(\mathsf{t}(G)) \simeq G \tag{1}$$

By doing so, we get that the graphs of treewidth at most two are exactly the ones that can be expressed using the syntax.

Our key contribution then consists in giving a finite equational axiomatisation of graph isomorphism over this syntax. This answers positively the question asked by Courcelle and Engelfriet in their book, for treewidth two [13, p. 118].

Note that the choice of the syntax is important. Various finite syntaxes have already been proposed [15, 16, 13] to capture graphs of treewidth at most k, for a given k. However, some choices prevent finite presentations. For instance, while the *converse* operation we use could be eliminated by pushing it to the leaves, doing so would turn some of our axioms into infinite equational schemes. (See also Remark 27.)

As explained above, a few laws are rather natural like associativity of the two compositions, commutativity of $\parallel$, or the facts that 1 and $\top$ are neutral elements and that $\cdot^\circ$ is an involution. Those are the first eight laws in Figure 3. Surprisingly, the four subsequent axioms suffice to obtain a complete axiomatisation: for all terms u, v, u and v are provably equal using the axioms from Figure 3 if and only if $\mathsf{g}(u)$ and $\mathsf{g}(v)$ are isomorphic:

$$u \equiv v \quad \Leftrightarrow \quad \mathsf{g}(u) \simeq \mathsf{g}(v) \tag{2}$$

In other words, calling a *2p-algebra* an algebra satisfying the axioms from Figure 3, TW_2 is the free 2p-algebra.

All axioms but (A3) are independent; (A3) follows from (A9) and (A12). Correctness, i.e., the left-to-right implication in (2), is easy to establish. Indeed, it suffices to compute and compare the graphs of each equation, and to prove that valid equations are stable under graph substitution. The converse implication, completeness, is much harder. This is because there is no canonical way of extracting a term out of a graph. In particular, the function t we define to this end has to make choices based on the concrete representation of the input graph, so that isomorphic graphs do not always map to syntactically equal terms.

[1] This additional condition is natural when considering pointed graphs [11]; this is not a restriction for unpointed graphs as one can always set input and output to the same arbitrary node.

$$u \parallel (v \parallel w) \equiv (u \parallel v) \parallel w \quad \text{(A1)} \qquad u{\cdot}(v{\cdot}w) \equiv (u{\cdot}v){\cdot}w \quad \text{(A4)}$$

$$u \parallel v \equiv v \parallel u \quad \text{(A2)} \qquad u{\cdot}1 \equiv u \quad \text{(A5)}$$

$$u \parallel \top \equiv u \quad \text{(A3)}$$

$$1 \parallel 1 \equiv 1 \quad \text{(A9)}$$

$$u^{\circ\circ} \equiv u \quad \text{(A6)} \qquad 1 \parallel u{\cdot}v \equiv 1 \parallel (u \parallel v^{\circ}){\cdot}\top \quad \text{(A10)}$$

$$(u \parallel v)^{\circ} \equiv u^{\circ} \parallel v^{\circ} \quad \text{(A7)} \qquad u{\cdot}\top \equiv (1 \parallel u{\cdot}\top){\cdot}\top \quad \text{(A11)}$$

$$(u{\cdot}v)^{\circ} \equiv v^{\circ}{\cdot}u^{\circ} \quad \text{(A8)} \qquad (1 \parallel u){\cdot}v \equiv (1 \parallel u){\cdot}\top \parallel v \quad \text{(A12)}$$

Figure 3 Twelve axioms for 2p-algebras, all independent except for (A3)

We proceed in the following way to obtain completeness. First we prove that the function t maps isomorphic graphs to congruent terms:

$$G \simeq H \qquad \Rightarrow \qquad \mathsf{t}(G) \equiv \mathsf{t}(H) \tag{3}$$

Then we prove that this function is a homomorphism (up to the axioms), which allows us to deduce that for all terms u, $\mathsf{t}(\mathsf{g}(u))$ is provably equal to u:

$$\mathsf{t}(\mathsf{g}(u)) \equiv u \tag{4}$$

In a sense, by interpreting a term u into a graph and then reading it back, we obtain a term $\mathsf{t}(\mathsf{g}(u))$ which plays the role of a normal form even if it is not canonical (which would typically be the case in rewriting theory, or in normalisation by evaluation [4]).

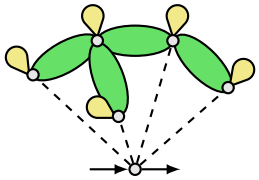
Defining a function t satisfying (1) could be done rather easily by relying on the notion of tree decomposition. However, doing so makes it extremely difficult to obtain properties (3) and (4): the notion of tree decomposition, despite its inductive nature, does not provide enough structure. Instead, we use the fact that treewidth at most two graphs are K_4-free, and we exhibit stronger graph invariants that allow us to extract terms from graphs in a much more structured way.

For instance, when the graph is connected and when its input and output are distinct, one can compute its *checkpoints*: those vertices which all paths from the input to the output must visit. Those checkpoints are linearly ordered so that the graph has the following shape

(5)

If there is at least one checkpoint then the graph should be interpreted as a series composition. Otherwise, by the absence of K_4 as a minor, one can show that the graph necessarily is a non-trivial parallel composition.

The aforementioned step is already there in the standard result that the biconnected components of a K_4-free graph are series-parallel. More challenging is the case when the input and output coincide. In this case, we consider the checkpoints of all pairs of neighbours of the input, and we show that they form a tree which is a minor of the starting graph.



This tree is a key invariant of the isomorphism class of the graph and we show that one can extract a term for each choice of a node in this tree. This is where our function $\mathtt{t}$ has to rely on the concrete representation of the graph: although all choices of a node in the tree result in provably equal terms, they do not yield syntactically equal terms. A similar situation happens with components which are disconnected from the input and the output: we handle those recursively by taking any vertex as a new choice of input and output.

2 Related work

Except for the presence of $\top$, the algebra of graphs we work with has been proposed independently by Freyd and Scedrov [20, p. 207], and by Andréka and Bredikhin [1]. They used it to characterise the equational theory of binary relations over the considered signature. Indeed the set of binary relations over a fixed set forms an algebra for the signature we consider in this paper: $\cdot$ is relational composition, $\parallel$ is set-theoretic intersection (thus it is written $\cap$ in [20, 1]), $\cdot^\circ$ is transposition, 1 is the identity relation and $\top$ is the full relation. Writing $\mathrm{Rel} \vDash u \leq v$ for the containments that hold in all such algebras of relations, and $G \blacktriangleleft H$ if there exists a graph homomorphism from H to G, we have the following equivalence.

$$\mathrm{Rel} \vDash u \leq v \qquad \Leftrightarrow \qquad \mathsf{g}(u) \blacktriangleleft \mathsf{g}(v)$$

This characterisation immediately gives decidability: existence of a graph homomorphism is an NP-complete problem. Thanks to the present observation that graphs of terms have bounded treewidth, the complexity is actually polynomial [21].

Freyd and Scedrov also use this characterisation to prove that this theory is not finitely presentable [20]: every complete equational axiomatisation must contain axioms corresponding to homomorphisms equating arbitrarily many vertices at a time, and thus must be infinite. Andréka and Bredikhin go even further and show that it is not even a variety [1].

In this work we focus on isomorphism rather than on homomorphism, and this is why we do obtain a finite equational axiomatisation. Although all algebras of relations validate our axioms, these algebras cannot be free models. For instance, their parallel composition (intersection) is always idempotent. Freyd and Scedrov remark that certain graphs cannot be the image of a term [20, p. 207], and Andréka and Bredikhin use a weak form of the K_4 exclusion property [1, Lemma 7]. They cannot obtain a characterisation result since they do not consider $\top$, which is necessary to reach all graphs of treewidth at most two.

Our work is also really close to that of Dougherty and Gutiérrez [17], who proposed an axiomatisation of graph isomorphism for a slightly different syntax: instead of the constant $\top$, they use a unary operation $\mathrm{dom}(\cdot)$, called *domain*. This operation can be defined in our setting: we have $\mathrm{dom}(u) = 1 \parallel (u \cdot \top)$; at the graphical level, it consists in relocating the output of a graph on its input. In contrast, $\top$ cannot be defined in terms of $\mathrm{dom}(\cdot)$ and the other operations. Choosing this domain operation has the advantage of keeping connected graphs, and the disadvantage of being less general: disconnected graphs cannot be expressed. More importantly, the operation $\top$ being more primitive than $\mathrm{dom}(\cdot)$, we can obtain a shorter axiomatisation: while we share with [17] the nine natural axioms from Figure 3 that do not mention $\top$, the four remaining ones in this figure have to be replaced by nine axioms when using $\mathrm{dom}(\cdot)$: three about 1 and $\parallel$, and six about $\mathrm{dom}(\cdot)$. To prove completeness, Dougherty and Gutiérrez compute normal forms for terms using rewriting techniques. Like in the present work, their normal forms are not canonical and some additional work is needed. In a second part of the paper, they characterise graphs of terms using a minor exclusion theorem which corresponds precisely to what we obtain in

the connected case (see Remark 27). There are however several typos or gaps in their paper which we were not able to fix—see [8] for more details.

Bauderon and Courcelle gave a syntax and a complete axiomatisation for arbitrary graphs [3]. While the overall statement is similar to ours, their syntax can hardly be related to the present one (it is infinitary, for instance), and the present results are not corollaries of their work. The structural invariants we exhibit here are reminiscent of the general decomposition results of Tutte [27], which Courcelle later studied in the context of MSO [12].

3 2p-algebra

We consider the signature $\langle \cdot_2, \parallel_2, \cdot^\circ_1, 1_0, \top_0 \rangle$ and we let u, v, w range over *terms* over a set Σ of variables. We usually omit the $\cdot$ symbol and we assign priorities so that the term $(a \cdot (b^\circ)) \parallel c$ can be written just as $ab^\circ \parallel c$. A *2p-algebra* is an algebra over this signature satisfying the axioms from Figure 3. We write $u \equiv v$ when two terms u and v are congruent modulo those axioms, or equivalently, when the equation holds in all 2p-algebras.

Following notations from Kleene algebra with tests (KAT) [22], we let α, β range over *tests*, those terms that are congruent to some term of the shape $1 \parallel u$. (By axiom A9, u is a test iff $u \equiv 1 \parallel u$.) Graphs of tests are those whose input and output coincide.

We shall use the derived operation mentioned in Section 2, *domain*, as well as its dual, *codomain*: $\mathrm{dom}(u) \triangleq 1 \parallel u\top$ and $\mathrm{cod}(u) \triangleq 1 \parallel \top u$. Those are tests by definition.

As is standard for involutive monoids, the first eight axioms from Figure 3 entail $1^\circ \equiv 1$, $\top^\circ \equiv \top$, and $1u \equiv u$. We use such laws freely in the sequel. We recall the four remaining axioms below, using the above notations.

$$1 \parallel 1 \equiv 1 \qquad \text{(A9)} \qquad\qquad u\top \equiv \mathrm{dom}(u)\top \qquad \text{(A11)}$$

$$1 \parallel uv \equiv \mathrm{dom}(u \parallel v^\circ) \qquad \text{(A10)} \qquad\qquad \alpha v \equiv \alpha\top \parallel v \qquad \text{(A12)}$$

Thanks to converse being an involution, there is a notion of duality in 2p-algebras: one obtains a valid law when swapping the arguments of all products and exchanging domains with codomains in a valid law. (We have $\mathrm{cod}(u) \equiv \mathrm{dom}(u^\circ)$.)

▶ **Proposition 1.** *The following equations hold in all 2p-algebras.*

$$\alpha^\circ \equiv \alpha \qquad (6)$$

$$\alpha\beta \equiv \alpha \parallel \beta \qquad (7)$$

$$\alpha(v \parallel w) \equiv \alpha v \parallel w \qquad (8)$$

$$(v \parallel w)\alpha \equiv v\alpha \parallel w \qquad (9)$$

$$\mathrm{dom}(uv \parallel w) \equiv \mathrm{dom}(u \parallel wv^\circ) \qquad (10)$$

$$\top u^\circ \top \equiv \top u \top \qquad (11)$$

$$u\top w \equiv u\top \parallel \top w \qquad (12)$$

$$u\top v\top w \equiv u\top w \parallel \top v\top \qquad (13)$$

$$(u \parallel \top v\top)w \equiv uw \parallel \top v\top \qquad (14)$$

4 Graphs

As explained in the introduction, we consider labelled directed graphs with two designated vertices. We just call them graphs in the sequel. Note that we allow multiple edges between two vertices, as well as self-loops.

▶ **Definition 2.** A *graph* is a tuple $G = \langle V, E, s, t, l, \iota, o \rangle$, where V is a finite set of *vertices*, E is a finite set of *edges*, $s, t : E \to V$ are maps indicating the *source* and *target* of each edge, $l : E \to \Sigma$ is map indicating the *label* of each edge, and $\iota, o \in V$ are the designated vertices, respectively called *input* and *output*.

We write $G[x;y]$ for the graph G with input set to x and output set to y; we abbreviate $G[x;x]$ to $G[x]$.

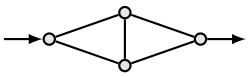
▶ **Definition 3.** An *homomorphism* from $G = \langle V, E, s, t, l, \iota, o\rangle$ to $G' = \langle V', E', s', t', l', \iota', o'\rangle$ is a pair $h = \langle f, g\rangle$ of functions $f : V \to V'$ and $g : E \to E'$ that respect the various components: $s' \circ g = f \circ s$, $t' \circ g = f \circ t$, $l' = g \circ l$, $\iota' = f(\iota)$, and $o' = f(o)$.

A *(graph) isomorphism* is a homomorphism whose two components are bijective functions. We write $G \simeq G'$ when there exists an isomorphism between graphs G and G'.

▶ **Proposition 4.** *Graphs up to isomorphism form a 2p-algebra.*

▶ **Definition 5.** Let $G = \langle V, E, s, t, l, \iota, o\rangle$ be a graph. A *tree decomposition* of G is a tree T where each node t is labelled with a set $V_t \subseteq V$ of vertices, such that:
(T1) for every vertex $x \in V$, the set of nodes t such that $x \in V_t$ forms a sub-tree of T;
(T2) for every edge $e \in E$, there exists a node t such that $\{s(e), t(e)\} \subseteq V_t$;
(T3) there exists a node t such that $\{\iota, o\} \subseteq V_t$.
The *width* of a tree decomposition is the size of the largest set V_t minus one; the *treewidth* of a graph is the minimal width of a tree decomposition for this graph. We write TW_2 for the set of graphs of treewidth at most two.

The first two conditions in the definition of tree decomposition are standard; the third one is related to the presence of distinguished nodes: it requires them to lie together in some node of the tree. This condition ensures that the following graph is excluded from TW_2 whatever the orientation and labelling of its edges.



(M_3)

Indeed, such a graph cannot be represented in the syntax we consider. (Something already observed by Freyd and Scedrov [20]—the addition of $\top$ to the syntax does not help.)

▶ **Proposition 6.** *Graphs of treewidth at most two form a subalgebra of the algebra of graphs.*

The graphs we associate to each letter (Figure 2) also belong to this subalgebra, so that we obtain a homomorphism $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$ associating a graph of treewidth at most two to each syntactic term. When taking quotients under term congruence and graph isomorphism, this function becomes a 2p-algebra homomorphism $\mathsf{g}' : \mathsf{Trm}_{/\equiv} \to \mathsf{TW}_{2/\simeq}$. Our key result is that g' actually is an isomorphism of 2p-algebras (Corollary 33).

5 K_4-freeness

In this section we establish preliminary technical results about unlabelled undirected graphs with at most one edge between two vertices and without self-loops; we call those *simple graphs*. We use standard notation and terminology from graph theory [16]. In particular, we denote by xy a potential edge between two vertices x and y; an xy-path is a (possibly trivial) path whose ends are x and y; $G + xy$ is the simple graph obtained from G by adding the edge xy if x and y were not already adjacent; $G \backslash x$ is the simple graph obtained from G by removing the vertex x and its incident edges.

▶ **Definition 7.** A *minor* of a simple graph G is a simple graph obtained by performing a sequence of the following operations on G: delete an edge or a vertex, contract an edge.

A cornerstone result of graph theory, Robertson and Seymour's graph minor theorem [26], states that (simple) graphs are well-quasi-ordered by the minor relation. As a consequence, the classes of graphs of bounded treewidth, which are closed under taking minors, can be characterised by finite sets of excluded minors. Two simple and standard instances are the following ones: the graphs of treewidth at most one (the forests) are precisely those excluding the cycle with three vertices (C_3); those of treewidth at most two are those excluding the complete graph with four vertices (K_4) [18]. We eventually reprove the latter one here.

 (C_3)  (K_4)

We fix a connected simple graph G in the remainder of this section.

▶ **Definition 8.** The *checkpoints* between two vertices x, y are the vertices which any xy-path must visit: $\mathsf{CP}(x, y) \triangleq \{z \mid \text{every } xy\text{-path crosses } z\}$.

For all vertices x, y, we have $\mathsf{CP}(x, x) = \{x\}$ and $\{x, y\} \subseteq \mathsf{CP}(x, y) = \mathsf{CP}(y, x)$. Two vertices x, y are *linked*, written $x \Diamond y$, when $x \neq y$ and $\mathsf{CP}(x, y) = \{x, y\}$, i.e., when there are no proper checkpoints between x and y. The *link graph* of G is the graph of linked vertices. Note that G is a subgraph of its link graph: if xy is an edge in G then $x \Diamond y$. We also have the following properties.

▶ **Lemma 9.** *Any cycle in the link graph is actually a clique.*

▶ **Lemma 10.** *If xyz is a triangle in the link graph and ι is a vertex not in G, then the graph $G + \iota x + \iota y + \iota z$ admits K_4 as a minor.*

Now fix a set U of vertices; we extend the notion of checkpoints as follows.

▶ **Definition 11.** The *checkpoints* of U, $\mathsf{CP}(U)$, is the set of vertices which are checkpoints of some pair in U: $\mathsf{CP}(U) \triangleq \bigcup_{x,y \in U} \mathsf{CP}(x, y)$. The *checkpoint graph* of U is the subgraph of the link graph induced by this set. We also denote this graph by $\mathsf{CP}(U)$.

▶ **Lemma 12.** CP *is a closure operator on the set of vertices. In particular, for all checkpoints $x, y \in \mathsf{CP}(U)$, $\mathsf{CP}(x, y) \subseteq \mathsf{CP}(U)$.*

▶ **Lemma 13.** *For every path in G between two checkpoints $x, y \in \mathsf{CP}(U)$, the sequence obtained by keeping only the elements in $\mathsf{CP}(U)$ is an xy-path in $\mathsf{CP}(U)$.*

Since G is assumed to be connected, it follows that so is $\mathsf{CP}(U)$. A key instance of a checkpoint graph is when U only contains two vertices, presumably the input and output of some graph: the checkpoint graph is a line in this case, as in (5), and it allows us to decompose the considered graph into a sequence of series compositions.

▶ **Lemma 14.** *If $U = \{x, y\}$, then $\mathsf{CP}(U)$ is a line graph whose ends are x and y.*

The following two lemmas are helpful in Proposition 20 below, to prove that the checkpoint graph is a tree under certain circumstances.

▶ **Lemma 15.** *If xy is an edge in $\mathsf{CP}(U)$, then there exists $x', y' \in U$ such that x and y belong to $\mathsf{CP}(x', y')$.*

▶ **Lemma 16.** *If xyz is a triangle in $\mathsf{CP}(U)$, then there exists $x', y', z' \in U$ such that x and y (resp. x and z, y and z) belong to $\mathsf{CP}(x', y')$ (resp. $\mathsf{CP}(x', z')$, $\mathsf{CP}(y', z')$).*

As explained above we use the checkpoint graphs to decompose graphs. The following notions of intervals and bags are the basic blocks of those decompositions.

▶ **Definition 17.** Let x, y be two vertices. The *strict interval* $]\!]x;y[\!]$ is the following set of vertices.

$$]\!]x;y[\!] \triangleq \{p \mid \text{there is an } xp\text{-path avoiding } y \text{ and a } py\text{-path avoiding } x\}$$

The *interval* $[\![x;y]\!]$ is obtained by adding x and y to that set. We abuse notation and write $[\![x;y]\!]$ for the subgraph of G induced by the set $[\![x;y]\!]$.

Note that while the intervals do not depend on the set U, we mostly use them under the assumption that xy is an edge in a checkpoint graph.

▶ **Definition 18.** The *bag* of a checkpoint $x \in \mathsf{CP}(U)$ is the set of vertices that need to cross x in order to reach the other checkpoints.

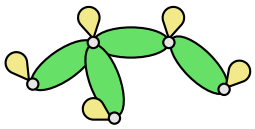
$$[\![x]\!]_U \triangleq \{p \mid \forall y \in \mathsf{CP}(U),\ \text{any } py\text{-path crosses } x\}\,.$$

As before, we also write $[\![x]\!]_U$ for the induced subgraph of G.

Note that $[\![x]\!]_U$ depends on U and differs from $[\![x;x]\!]$ (which is always the singleton $\{x\}$).

▶ **Proposition 19.** *If $\mathsf{CP}(U)$ is a tree, then the following set $\mathcal{V}$ is a partition of the vertices of G such that any edge of G appears in exactly one graph of the set $\mathcal{E}$.*

$$\mathcal{V} \triangleq \{[\![x]\!]_U \mid x \in \mathsf{CP}(U)\} \cup \{]\!]x;y[\!] \mid xy \text{ edge in } \mathsf{CP}(U)\}$$
$$\mathcal{E} \triangleq \{[\![x]\!]_U \mid x \in \mathsf{CP}(U)\} \cup \{[\![x;y]\!] \mid xy \text{ edge in } \mathsf{CP}(U)\}$$



Graphically, this means G can be decomposed as in the picture above; only the vertices of $\mathsf{CP}(U)$ are depicted, the green blocks correspond to edges in $\mathsf{CP}(U)$, the yellow blocks correspond to the graphs $[\![x]\!]_U$. The leaves of $\mathsf{CP}(U)$ are elements of U (but not always conversely). As a consequence, when $\mathsf{CP}(U)$ is a tree, it is a minor of G: contract all subgraphs of the form $[\![x]\!]_U$ into vertex x and all subgraphs of the form $[\![x;y]\!]$ into edge xy.

The following proposition is a key element in the developments to come. It makes it possible to extract a term out of a graph whose input and output coincide, by providing ways to chose an element where to relocate the output and resort to the easier case when input and output differ. (Note that G is still assumed to be connected.)

▶ **Proposition 20.** *Assume $G = H\backslash\iota$, for some K_4-free simple graph H and some vertex ι. Further assume that U is the set of neighbours of ι in H and that this set is not empty.*

(i) *$\mathsf{CP}(U)$ is a tree,*
(ii) *for every edge xy in $\mathsf{CP}(U)$, the graph $[\![x;y]\!] + xy$ is K_4-free,*
(iii) *for every vertex x in $\mathsf{CP}(U)$, the graph $H + \iota x$ is K_4-free.*

As a consequence of the above proposition, we have the following one, which makes it possible to decompose graphs with distinct input and output into a parallel composition when they cannot be a series composition.

▶ **Proposition 21.** *Let ι, o be two distinct vertices such that $G + \iota o$ is K_4-free. We have that:*

(i) *if ι and o are not adjacent in G and $\iota \Diamond o$, then the graph induced by $[\![\iota; o]\!]$ has at least two connected components.*

(ii) *for every edge xy in $\mathsf{CP}(\{\iota, o\})$, the graph $[\![x; y]\!] + xy$ is K_4-free,*

6 Extracting terms

Now we have enough preliminary material and we can look for a right inverse to the function $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$. As explained in the Introduction, we use K_4-freeness to extract terms from graphs in a more structured way than using tree decompositions directly.

▶ **Definition 22.** The *skeleton* of a graph G is the simple graph S obtained from G by forgetting input, output, labelling, edge directions, edge multiplicities, and self-loops. The *strong skeleton* of G is $S + \iota o$ if $\iota \neq o$, and S otherwise.

As an example, the strong skeleton of any instance of the graph (M_3) from Section 4 is K_4. More generally, a graph belongs to TW_2 if and only if its strong skeleton has treewidth at most two in the standard sense.

▶ **Proposition 23.** *The strong skeleton of every graph in TW_2 is K_4-free.*

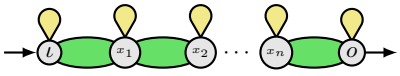
Given a graph G and two vertices x, y, we write $G[\![x; y]\!]$ for the subgraph of G induced by the set $[\![x; y]\!]$ (computed in the skeleton of G), with input and output respectively set to x and y, and with self-loops on x and y removed. The strong skeleton of $G[\![x; y]\!]$ is $[\![x; y]\!] + xy$.

Similarly, given a graph G, a set U of vertices and vertex x, we write $G[\![x]\!]_U$ for the subgraph of G induced by the set $[\![x]\!]_U$ (computed in the skeleton of G), with both input and output set to x. Doing so, the skeleton and strong skeleton of $G[\![x]\!]_U$ are both $[\![x]\!]_U$. We shall omit the subscript when it is clear from the context.

▶ **Definition 24.** The term $\mathsf{t}(G)$ of a graph G whose strong skeleton is K_4-free is defined by induction on the number of edges in G[2]. When G is connected there are two main cases depending on whether the input and output coincide *(a)* or not *(b)*. We deal with the general case *(c)* by decomposing the graph into connected components.

(a) Connected, distinct input and output

Consider the line graph (Lemma 14) obtained by taking the checkpoint graph of $U = \{\iota, o\}$ in the skeleton of G. Write it as $x_0 \dots x_{n+1}$ with $\iota = x_0$ and $o = x_{n+1}$. According to Proposition 19, G looks as follows.



We set $\mathsf{t}(G) \triangleq \mathsf{t}(G[\![x_0]\!]) \cdot \mathsf{t}(G[\![x_0; x_1]\!]) \cdot \mathsf{t}(G[\![x_1]\!]) \cdot \ldots \cdot \mathsf{t}(G[\![x_n]\!]) \cdot \mathsf{t}(G[\![x_n; x_{n+1}]\!]) \cdot \mathsf{t}(G[\![x_{n+1}]\!])$

The (strong) skeleton of each graph $G[\![x_i]\!]$ is just $[\![x_i]\!]$, which is necessarily K_4-free, as a subgraph of that of G. Proposition 21(2) moreover ensures that so are the strong skeletons of all graphs $G[\![x_i; x_{i+1}]\!]$. The above recursive calls occur on smaller graphs unless $n = 0$ and the graphs $G[\![\iota]\!]$ and $G[\![o]\!]$ are reduced to the trivial graph with one vertex and no edge (i.e., the graph 1). In such a situation,

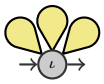
[2] More precisely, on the lexicographic product of the number of edges and the textual precedence of the three considered cases.

- either ι and o are adjacent in G. Then let G' be the graph obtained by removing from G all edges between ι and o and let $u = a_1 \parallel \ldots \parallel a_i \parallel b_1^\circ \parallel \ldots b_j^\circ$ be a term corresponding to those edges. Accordingly, we set $\mathsf{t}(G) \triangleq \mathsf{t}(G') \parallel u$.
- Or they are not, and Proposition 21(2) applies so that we can decompose G into parallel components: $G = G_1 \parallel \ldots \parallel G_m$ with $m \geq 2$. We set $\mathsf{t}(G) \triangleq \mathsf{t}(G_1) \parallel \ldots \parallel \mathsf{t}(G_m)$.

(b) Connected, input equals output

If there are self-loops on ι, let $u = a_1 \parallel \ldots \parallel a_n$ be a term corresponding to those edges, let G' be the graph obtained by removing them, and recursively set $\mathsf{t}(G) \triangleq \mathsf{t}(G') \parallel u$.

Otherwise let H be the skeleton of G. Decompose $H\backslash\iota$ into connected components $H_1\backslash\iota, \ldots, H_m\backslash\iota$ such that $H \simeq H_1 \cup \cdots \cup H_m$. The graph looks as follows.



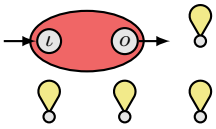
If $m = 0$, then set $\mathsf{t}(G) = 1$. If $m > 1$, set $\mathsf{t}(G) \triangleq \|_{i \leq m} \mathsf{t}(G_i)$, where G_i is the subgraph of G induced by H_i. It remains to cover the case where $m = 1$. Let U be the set of neighbours of the input and compute the checkpoint graph $\mathsf{CP}(U)$ in $H\backslash\iota$. Pick an arbitrary node $x \in \mathsf{CP}(U)$. By Proposition 20(3), the strong skeleton of $G[\iota; x]$ is K_4-free. Set $\mathsf{t}(G) \triangleq \mathrm{dom}(\mathsf{t}(G[\iota; x]))$. (Remember that $\mathrm{dom}(\cdot)$ relocates the output to the input.)

(c) General case

Decompose the graph G into connected components $G_1, \ldots, G_n$. For all $i \leq n$, pick an arbitrary vertex x_i in the component G_i. There are two cases:

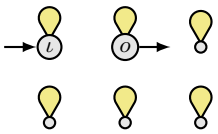
- either input and output belong to the same component, say G_j; then set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G_j) \parallel \Big\|_{i \neq j} \top \cdot \mathsf{t}(G_i[x_i]) \cdot \top$$



- or they belong to two distinct components, say ι in G_j and o in G_k, in which case we set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G_j[\iota]) \cdot \top \cdot \mathsf{t}(G_k[o]) \parallel \Big\|_{i \neq j,k} \top \cdot \mathsf{t}(G_i[x_i]) \cdot \top$$



In both cases, it is easy to check that the recursive calls occur on graphs whose (strong) skeletons are subgraphs of the strong skeleton of G, and thus K_4-free.

The definition of the extraction function t ends here. This function is defined on "concrete" graphs: we need to choose some vertices in cases *(b)* and *(c)*, and we can only do so by relying on the concrete identity of those vertices (e.g., choosing the smallest one, assuming they are numbers). We shall see in the following section that all those potential choices, nevertheless, always lead to congruent terms (Theorem 30). By construction, we obtain:

▶ **Theorem 25.** *For every graph $G \in \mathsf{TW}_2$, $\mathsf{g}(\mathsf{t}(G)) \simeq G$.*

▶ **Corollary 26.** *The following are equivalent for all graphs G:*

(i) *G has treewidth at most two;*
(ii) *the strong skeleton of G is K_4-free;*
(iii) *G is (isomorphic to) the graph of a term.*

▶ Remark 27. When G is connected, $\mathsf{t}(G)$ does not contain occurrences of $\top$ other than those that are implicit in our uses of $\mathrm{dom}(\cdot)$ in case *(b)*. Thus we obtain an alternative proof of Dougherty and Gutiérrez' characterisation [17, Section 4, Theorem 31] (their minor exclusion property is easily proved equivalent to ours—they do not mention treewidth).

Also note that we can easily avoid using 1 (but not $\mathrm{dom}(\cdot)$) when the graph does not contain self-loops and is not reduced to the trivial graph 1. When the graph does not contain self-loops and has distinct input and output, the construction can be modifed to produce terms without both 1 and $\mathrm{dom}(\cdot)$; the resulting construction becomes, however, less local, and we do not know how to use it to axiomatise the 1-free reduct of 2p-algebra.

7 Completeness of the axioms

We can finally prove that the axioms of 2p-algebras are complete w.r.t. graphs: they suffice to equate all terms denoting the same graph up to isomorphism. For lack of space, we present only the main steps. Proofs for this last part consist in detailed analyses of the term extraction function (t) through inductive arguments following its recursive definition, and using the laws from Proposition 1 to relate the extracted terms. All details are in [8].

We first prove that t maps isomorphic graphs to congruent terms. We need for that the following propositions.

▶ **Proposition 28.** *Let $G \in \mathsf{TW}_2$ be a graph with $\iota = o$, without self-loops on ι. Let S be its skeleton, and assume that $S\backslash\iota$ is connected. Let U be the neighbours of ι in G and consider the checkpoint graph of U in the skeleton of $S\backslash\iota$. For all checkpoints x, y, we have* $\mathrm{dom}(\mathsf{t}(G[\iota, x])) \equiv \mathrm{dom}(\mathsf{t}(G[\iota, y]))$.

▶ **Proposition 29.** *Let $G \in \mathsf{TW}_2$ be a connected graph. For all vertices x, y, we have* $\top\mathsf{t}(G[x])\top \equiv \top\mathsf{t}(G[y])\top$.

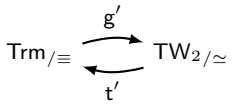
▶ **Theorem 30.** *Let $G, H \in \mathsf{TW}_2$ be two graphs. If $G \simeq H$ then* $\mathsf{t}(G) \equiv \mathsf{t}(H)$.

In other words, the extraction function t yields a function $\mathsf{t}' : \mathsf{TW}_{2/\simeq} \to \mathsf{Trm}_{/\equiv}$ between 2p-algebras. We finally prove that t' is a homomorphism, and, in fact, an isomorphism.

▶ **Proposition 31.** *The function $\mathsf{t}' : \mathsf{TW}_{2/\simeq} \to \mathsf{Trm}_{/\equiv}$ is an homomorphism of 2p-algebras.*

▶ **Theorem 32.** *For every term u, we have* $\mathsf{t}(\mathsf{g}(u)) \equiv u$.

▶ **Corollary 33.** *For all terms u and v, we have $u \equiv v$ if and only if $\mathsf{g}(u) \simeq \mathsf{g}(v)$. Graphs of treewidth at most two form the free 2p-algebra, as witnessed by the diagram on the right.*



8 Future work

What is the free idempotent 2p-algebra? (Where parallel composition is idempotent.) One could be tempted to switch to simple directed graphs, where there is at most one edge with a given label from one vertex to another. This is however not an option: the graphs of $ab \parallel ab$ and ab are not isomorphic. One could also consider equivalences on graphs that are weaker than isomorphism. The notion of (two-way) bisimilarity [25, 24] that come to mind does not work either: such an equivalence relation on graphs certainly validates idempotency of

parallel composition, but it also introduces new laws, e.g., $\top(1 \parallel aa)\top = \top(1 \parallel a)\top$, which are not even true in algebras of binary relations.

Courcelle used the algebraic theory he defined with Bauderon for arbitrary graphs [3] to propose a notion of graph recognisability [9], based on the generic framework by Mezei and Wright [23]. He proved that sets of graphs definable in MSO are recognisable. The converse does not hold in general. He later proved it for graphs of treewidth at most two [10] with a *counting* variant of MSO, conjecturing that it would be so for classes of graphs of bounded treewidth. This conjecture was proved only last year, by Bojańczyk and Pilipczuk [6].

The present work makes it possible to propose an alternative notion of recognisability for treewidth at most two, *2p-recognisability*: recognisability by a finite 2p-algebra. We conjecture that this notion coincides with recognisability. That recognisability entails 2p-recognisability is easy. The converse is harder; it amounts to proving that any finite congruence with respect to substitutions in treewidth at most two graphs can be refined into a finite congruence with respect to substitutions in arbitrary graphs. We see two ways of attaining this implication:

1. prove that 2p-recognisability entails MSO-definability, which could possibly be done along the lines of [10], by showing that our term extraction procedure is MSO-definable.
2. or use a slight generalisation of the result by Courcelle and Lagergren [14], relating recognisability to *k-recognisability* for graphs of treewidth at most k. Indeed, 2*p*-recognisability is really close to 2-recognisability. Unfortunately, Courcelle and Lagergren's result is established only for unlabelled, undirected graphs, without sources, while we need labelled directed graphs with two sources.

One can easily extend our syntax to cover graphs of treewidth at most k, with k sources, for a given k (see, e.g., [15, 2]). However, we do not know how to generate finite axiomatisations in a systematic way, for every such k. Moreover, our proof strategy heavily depends on the fact that when $k = 2$, K_4 is the only excluded minor. We would need another strategy to deal with the general case since the excluded minors are not known for $k \geq 4$.

References

1 H. Andréka and D. A. Bredikhin. The equational theory of union-free algebras of relations. *Algebra Universalis*, 33(4):516–532, 1995. doi:10.1007/BF01225472.

2 S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993. doi:10.1145/174147.169807.

3 Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20(2-3):83–127, 1987. doi:10.1007/BF01692060.

4 Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *LICS*, pages 203–211. IEEE, 1991. doi:10.1109/LICS.1991.151645.

5 H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

6 Mikołaj Bojańczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *LICS*, pages 407–416. ACM, 2016. doi:10.1145/2933575.2934508.

7 Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theoretical Computer Science*, 239(2):211–229, 2000. doi:10.1016/S0304-3975(99)00220-0.

8 Enric Cosme-Llópez and Damien Pous. K_4-free graphs as a free algebra, 2017. Full version of this extended abstract, with all proofs, available at https://hal.archives-ouvertes.fr/hal-01515752/. URL: https://hal.archives-ouvertes.fr/hal-01515752/.

9 B. Courcelle. The monadic second-order logic of graphs. I: Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

10 B. Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991. doi:10.1016/0304-3975(91)90387-H.

11 B. Courcelle. Recognizable sets of graphs: equivalent definitions and closure properties. *Mathematical Structures in Computer Science*, 4(1):1–32, 1994.

12 B. Courcelle. The monadic second-order logic of graphs XI: Hierarchical decompositions of connected graphs. *Theoretical Computer Science*, 224(1):35–58, 1999. doi:10.1016/S0304-3975(98)00306-5.

13 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.

14 B. Courcelle and J. Lagergren. Equivalent definitions of recognizability for sets of graphs of bounded tree-width. *Mathematical Structures in Computer Science*, 6(2):141–165, 1996. doi:10.1017/S096012950000092X.

15 Bruno Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In *Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1993. Proceedings of a Joint Summer Research Conference on Graph Minors held June 22 to July 5, 1991, at the University of Washington, Seattle. doi:10.1090/conm/147.

16 R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2005.

17 Daniel J. Dougherty and Claudio Gutiérrez. Normal forms for binary relations. *Theoretical Computer Science*, 360(1-3):228–246, 2006. doi:10.1016/j.tcs.2006.03.023.

18 R.J Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965. doi:10.1016/0022-247X(65)90125-3.

19 Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *NCAI*, pages 4–9. AAAI Press / The MIT Press, 1990. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-001.php.

20 P.J. Freyd and A. Scedrov. *Categories, Allegories*. North Holland. Elsevier, 1990. URL: https://books.google.fr/books?id=fCSJRegkKdoC.

21 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1:1–1:24, 2007. doi:10.1145/1206035.1206036.

22 D. Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997. doi:10.1145/256167.256195.

23 J. Mezei and J.B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11(1–2):3–29, 1967. doi:10.1016/S0019-9958(67)90353-1.

24 R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

25 David Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, pages 167–183, 1981. URL: http://dl.acm.org/citation.cfm?id=647210.720030.

26 Neil Robertson and P.D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

27 W. Tutte. *Graph Theory*. Addison-Wesley, Reading, MA, 1984.