ENS DE LYON

THESE

pour l'obtention du grade de Docteur, délivré par

l'ECOLE NORMALE SUPERIEURE DE LYON

Ecole Doctorale N°512
InfoMaths – Informatique et Mathématiques de Lyon

Discipline : INFORMATIQUE (Informatique)

Soutenue publiquement le 24 octobre 2025, par :

Samuel HUMEAU

Structures et équations de graphes sourcés: petite largeur arborescente et chemins disjoints.

Structures and equations of sourced graphs : small treewidth and disjoint paths.

Après avis de :

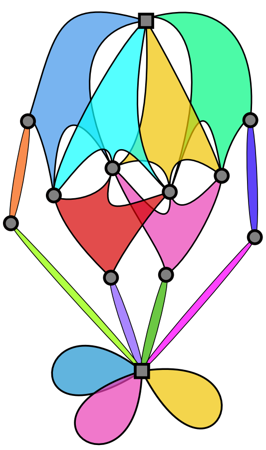
Marthe BONAMY, Directrice de recherche, Université de Bordeaux	Rapporteure
Mikołaj BOJAŃCZYK, Professeur, University of Warsaw	Rapporteur

Devant le jury composé de :

Marthe BONAMY, Directrice de recherche, Université de Bordeaux	Rapporteure
Mikołaj BOJAŃCZYK, Professeur, University of Warsaw	Rapporteur
Amina DOUMANE, Chargée de recherche, ENS de Lyon	Examinatrice
Radu IOSIF, Directeur de recherche, Université Grenoble Alpes	Examinateur
Ignasi SAU VALLS, Directeur de recherche, Université de Montpellier	Examinateur
Damien POUS, Directeur de recherche, ENS de Lyon	Directeur de thèse

Structures and Equations of Sourced Graphs: Small Treewidth and Disjoint Paths

Samuel Humeau



REMERCIEMENTS

Merci d'abord à Amina et Damien de m'avoir suivi, soutenu, et encadré depuis le printemps 2021, plus d'un an avant le début de mes années de doctorant. Nos interactions, toujours bienveillantes, ont été des plus fructueuses ! Ce manuscrit n'aurait pas pu exister sans vos suggestions, vos critiques, vos idées, votre confiance...

Aux quatre membres extérieurs de mon jury, Marthe, Mikołaj, Ignasi et Radu, merci d'avoir bien voulu me lire et m'écouter. C'est avec une profonde reconnaissance que je lis vos noms sur la page de garde de ce manuscrit.

À tous les anciens et actuels membres du LIP que j'ai côtoyés, les souvenirs des pauses, au coin café ou sur les balcons, des enseignements partagés, des galères administratives, et de tous les autres joyeux moments resteront longtemps avec moi. Je pense en particulier à Théophile, Pierre, Peio, Calvin, Hugo, mais aussi aux MALIPs et à Marie, dont l'efficacité ne souffre aucune faille, et aux membres de mon équipe : je n'ai que trop peu partagé votre science PLUMesque. Merci aussi à Jurriaan de m'avoir acceuilli à Nijmegen (prononcé "Naïmérreine") dans la poursuite et la fin de mes travaux de master. C'est toujours un plaisir de t'y trouver.

Enfin, à toutes celles et tous ceux qui m'ont soutenu pendant ces trois années, voire qui ont accepté de m'écouter quelques minutes parler de mes diableries de graphes, merci d'avoir été là et de l'être encore. Pour les curieux, voici une liste non exhaustive (les homonymes ne sont cités qu'une seule fois): Léna, Jérémy, Paul, Quentin, Noémie, Lucas, Marianne, Chiara, encore Léna, Joséphine, Rayan, Clair, Albane, Antoine, tous les autres membres du camping, de la coloc de Domène puis Voiron, l'ancienne équipe de la K-Fêt, mes parents et beaux-parents, mes frères et leurs compagnes, et enfin mon neveu et plus récemment ma nièce, dont les naissances encadrent presque parfaitement mes trois années en doctorat.

RÉSUMÉ

Une décomposition arborescente permet de comprendre un graphe quelconque comme un arbre, au prix de l'identification de certains sommets. En minimisant le nombre maximal de sommets identifiés entre eux, on obtient la largeur arborescente d'un graphe. Il s'agit là d'un paramètre de graphe exprimant la proximité d'un graphe à une structure d'arbre.

La largeur arborescente a de multiples origines et de nombreuses applications. Parmi ces différentes caractérisations on peut compter celle par les mineurs exclus minimaux, et celle par des systèmes de réécriture de graphes. Une des particularités de ces caractérisations est que les mineurs et systèmes de réécriture exacts ne sont pas connus dès que la largeur arborescente dépasse 4 ou 5, respectivement.

Une autre caractérisation est la suivante: les graphes de largeur arborescente au plus k sont générés par une syntaxe inductive finie dont les opérations sont interprétées sur les graphes. C'est une généralisation naturelle de la définition inductive des arbres: un sommet isolé est un arbre dont il est la racine, un arbre est une racine adjacente aux racines de ses sous-arbres. Cette syntaxe force une notion d'enracinement. Pour la largeur arborescente, en place d'une racine unique, un graphe peut avoir jusqu'à k+1 racines, appelées sources et servant d'interface pour les opérations de graphes.

Un problème posé par Courcelle et Engelfriet est le suivant: existe-t-il une théorie équationnelle finie correspondant à l'isomorphisme des graphes de largeurs arborescente bornée ? Autrement dit, existe-t-il une liste finie d'axiomes équationnels sur la syntaxe des graphes de largeur au plus k de sorte que deux termes associés soit prouvablement égaux si, et seulement si, leur graphes sont isomorphes ? Pous et Cosme-Llopez ont résolu le cas k=2 par la positive. La première partie de cette thèse donne une réponse positive au cas k=3.

On utilise alors une syntaxe généralisant celle des graphes série-parallèles. On introduit des notions de connectivité pour les graphes de largeur bornée (correspondant au composantes connexes, points d'articulation et 2-séparateurs, respectivement). On utilise ces outils pour analyser la structure des graphes de largeur au plus 3, montrant comment ceux-ci peuvent se décomposer récursivement, d'une manière d'abord canonique puis non-déterministe. La difficulté réside dans la preuve que tous les choix non-déterministes sont équivalents par les axiomes.

Notre preuve utilise un lemme décrivant la forme des graphes n'ayant pas de triangle comme mineur sourcé. Pour la généraliser à k=4, il faudrait un résultat similaire pour les graphes ne contenant pas de tétrahèdre comme mineur sourcé. Un tel résultat existe et sa preuve utilise le théorème des deux chemins.

Dans un graphe, un quadruplet (s,t,u,v) de sommets est 2-lié quand il existe deux chemins disjoints, l'un allant de s à t, et l'autre de u à v. Un graphe est dit 2-lié quand tout quadruplet de sommets est 2-lié.

La deuxième partie de cette thèse donne une nouvelle preuve simple du théorème des deux chemins. Ce dernier est une caractérisation par les toiles des graphes maximaux (vis-à-vis des arrêtes) n'étant pas 2-liés. Les toiles sont des graphes semblables à des triangulations du plans remplies de graphes complets. Notre preuve n'utilise ni le théorème de Menger, ni celui de Kuratowski. À la place, elle suit une définition inductive des toiles par des compositions parallèles. Cette opération est aussi l'une de celles générant les graphes de largeur bornée.

De notre preuve du théorème des deux chemins nous déduisons un algorithme récursif simple pour le problème des chemins disjoints. Cet algorithme est constructif en ce qu'il renvoie soit deux chemins disjoints, soit un plongement du graphe en entrée dans une toile.

ABSTRACT

A tree decomposition allows one to understand a graph as a tree via the identification of some vertices. By minimising the maximal number of vertices identified as a single vertex in such decompositions, we obtain the treewidth of the graph: this is a graph parameter somehow expressing how close a graph is to being a tree.

Treewidth has multiple origins as well as many applications. There are several equivalent definition of this parameter: a graph has treewidth at most k iff it avoids a certain finite list of graphs as minors, or iff it can be reduced to the empty graph via a certain graph rewriting system. The issue with these definitions is that the exact lists of minors or rewriting systems are not known explicitly whenever k is as big as 4 or 5, respectively.

Another characterisation of treewidth is the following: the graphs of treewidth at most k are generated by an inductive finite syntax whose operations are interpreted on graphs. This is a natural generalisation of the usual inductive characterisation of trees: an isolated vertex is a tree of which it is its root, a tree is a root adjacent to the roots of its subtrees. This syntax imposes a notion of rootedness. For treewidth, istead of a unique root, a graph has up to k+1 roots at a given time, called its sources and used as interface in graph operations.

A question asked by Courcelle and Engelfriet is the following: is there a finite equational theory corresponding to the isomorphism of graphs of treewidth at most k? Said otherwise, is there a finite list of equations on the syntax of graphs of treewidth at most k so that two terms from the syntax are provably equal via the axioms iff their graphs are isomorphic ? Pous and Cosme-Llopez have solved the case k=2 positively. The first part of this thesis has been focused on finding a positive proof for the case k=3.

For this proof, we use a syntax generalising series-parallel expressions, denoting graphs with a small interface. We introduce appropriate notions of connectivity for graph of bounded treewidth (corresponding to components, cutvertices, and 2-separators, respectively). We use those concepts to analyse the structure of graphs of treewidth at most 3, 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.

This proof relies on a specific lemma expressing which constraints apply on graphs avoiding the triangle as a minor. To generalise the proof to k=4, this lemma should be generalised to understand graphs avoiding the tetrahedron as a minor. Such a result already exists. Its proof relies on the two-paths theorem.

A tuple (s,t,u,v) of vertices in a simple undirected graph is 2-linked when there are two vertex-disjoint paths respectively from s to t and from u to v. A graph is 2-linked when all such tuples are 2-linked.

The second part of this thesis is a new and simple proof of the two-paths theorem, a characterisation of edge-maximal graphs which are not 2-linked as webs: particular near triangulations of the plane filled with cliques. Our proof does not require major theorems such as Kuratowski's or Menger's theorems. Instead it follows an inductive characterisation of generalised webs via parallel composition, a graph operation consisting in taking a disjoint union before identifying some pairs of vertices. This operation is also used in the operational characterisations of graphs of bounded treewidth discussed above.

We use the insights provided by this proof to design a simple O(nm) recursive algorithm for the "two vertex-disjoint paths" problem. This algorithm is constructive in that it returns either two disjoint paths, or an embedding of the input graph into a web.

Contents

Introduction

The main concern of this thesis is the study of *finite equational axiomatisations* of *graphs of bounded treewidth*, in particular for graphs of treewidth at most three. By studying finite equational axiomatisations, we dive into the structure of the corresponding objects. In doing so, we establish several known and novel structural results.

EQUATIONS AND BOUNDED TREEWIDTH

1. Treewidth. *Treewidth* is a graph parameter expressing how close a graph is to being a tree. It is common in parameterised complexity: many NP-complete problems become polynomial or even linear once parameterised using treewidth [7]; it is also a core concept of the graph minor theory of Robertson and Seymour, and one of the main tools in the proof of their celebrated theorem [32].

Treewidth admits many characterisations. It was introduced at least by Bertelè and Brioschi [6], Halin [18], and Robertson and Seymour [33]. Graphs of treewidth at most k for a non-negative integer k were subsequently characterised using k-trees [2], k-elimination graphs [49, 37, 35], and chordal graphs [33].

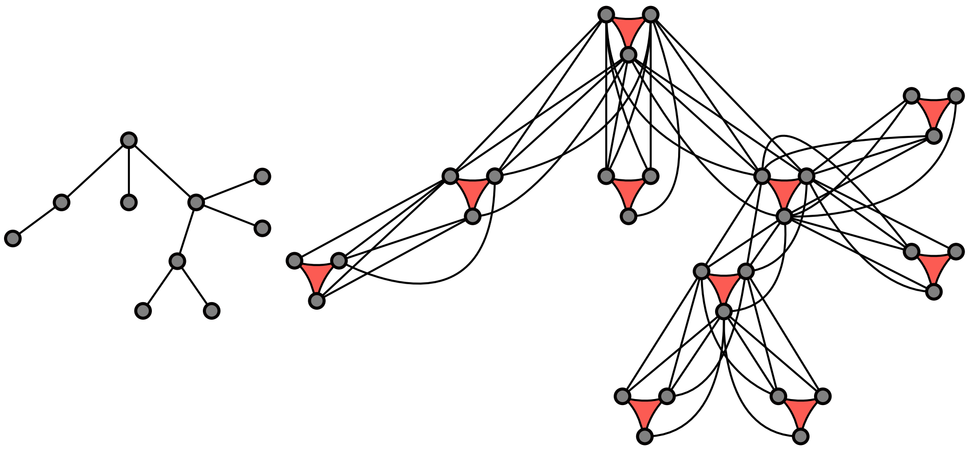


Figure 1: A tree and a 3-tree. The red triangles in the 3-tree represent cliques of size three corresponding to the vertices in the tree. Hence the 3-tree is a "thickened version" of the tree.

The characterisation via *k-trees* makes the relation between trees and treewidth visual. For example we depict a tree and a 3-tree sharing a similar structure in Figure 1.

Trees are built from a single vertex by repeatedly adding *leaves*, that is vertices of degree one. In contrast, a k-tree is built from a clique on k vertices by repeatedly adding *k-leaves*, that is vertices of degree k whose neighbours form a clique. A graph has treewidth at most k if and only if it is a subgraph of a k-tree. In particular, a graph has treewidth at most one if and only if it is a forest, that is a disjoint union of trees.

Treewidth also has structural characterisations. For every non-negative integer k, there exists a finite list M_k of graphs such that a graph G has treewidth at most k if and only if G does not contain graphs in M_k as minors [32]. Similarly, for each non-negative integer k, there exists a finite graph rewriting system R_k and a set of accepted graphs A_k such that the rewriting rules of R_k rewrite a graph G to $A \in A_k$ if and only if G has treewidth at most k [1].

Explicit values for M_k and R_k are only known for $k \leq 3$ and $k \leq 4$ respectively.

2. Terms and axioms. To characterise treewidth we may also consider syntaxes generalising series-parallel expressions [1, 11]. The idea is to use *terms* to denote graphs. An important problem is to understand when two terms denote the same graph. To this end, Courcelle and Engelfriet gave an *equational axiomatisation* for arbitrary graphs [12, p. 117]: a structured set of *equations*, or *axioms*, from which it is possible to equate all terms denoting a given graph.

Courcelle and Engelfriet prove that finite fragments of the syntax make it possible to capture precisely the graphs up to a given treewidth, while the question of axiomatisation completeness is left open [12, p. 118]: for a given non-negative integer k, is there a finite equational axiomatisation 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 [10, 14].

In this thesis, we give a positive answer for $k = 3$ (Theorem 5.2.1).

There are several motivations for seeking complete finite axiomatisations of graph isomorphisms.

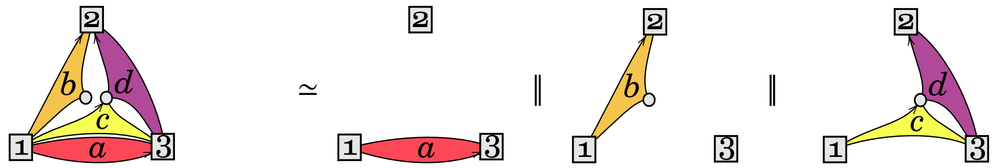
A first motivation is to provide compositional means to present (or look for) isomorphisms, via equational reasoning. Such tools are complementary to the various methods available for graphs of bounded treewidth [3, 28].

A second motivation arises from the algebraic theory of regular languages: the regular languages of graphs of bounded treewidth are those recognisable by homomorphisms into finite algebras, and finite axiomatisations make it possible to present such homomorphisms as finite objects (a finite algebra satisfying the axioms, together with the value of the homomorphism on letters).

In both cases the fundamental concept is *compositionality*: finite lists of axioms make it possible to summarise and delimit the interactions between the operations under consideration on graphs.

Another important motivation is of a graph-theoretical nature: finding a finite axiomatisation and proving its completeness forces us to analyse the structure of graphs of bounded treewidth in great depth, and leads us to new structural results. We believe that solving the problem for larger values of k, or for the general case, would also result in a better understanding of these classes of graphs.

Finally, the link between finite axiomatisations and the structure of graphs of bounded treewidth can be explained in the following manner. While our completeness result formally rests on a specific choice of syntax for terms, it may also be understood directly in terms of tree decompositions. Indeed, a term is roughly speaking a canonical tree decomposition in which each edge of the graph is associated to a bag containing its incident vertices. In other words, relating the parsings of a given graph is analogous to relating its different tree decompositions. Requiring a finite number of axioms then amounts to ensuring that tree decompositions can be transformed into one another via a finite set of local rules.



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

Figure 2: Two parsings of a given graph.

To prove the existence of a complete finite axiomatisation for graphs of treewidth at most three, we provide a syntax whose image under an interpretation map $\mathcal{G}$ from terms to graphs is exactly the set of graphs of treewidth at most three; and we give a finite list of axioms whose equational theory ($\equiv$) characterises graph isomorphism ($\simeq$). In symbols, we prove that for all terms t, l in the syntax of graphs of treewidth at most three,

$$\mathcal{G}(t) \simeq \mathcal{G}(l) \quad \text{if and only if} \quad t \equiv l$$

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* ($G \parallel H$): glue the graphs G and H 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).

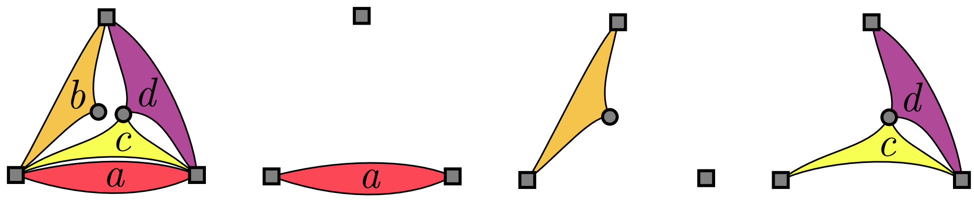
This syntax is similar though distinct from the one used by Arnborg, Courcelle, Proskurowski and Seese.

We call *arity* of a graph its number of sources, and *interface* its list of sources.

Consider the four graphs depicted in Figure 2, each with three sources denoted with numbered squares. The vertices incident to each edge are ordered, which we indicate by drawing an arrow from the first to the second vertices. 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 to 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 obtained from the decompositions of a graph using parallel compositions, permutations, lifts, and forgets as above.

Often, the specific order of sources, or of vertices incident to a given edge, is irrelevant (up to some permutation) or easily retrieved. We avoid specifying such orders in illustrations; hence the graphs in Figure 2 become:



The way we parsed the graph on the right in Figure 2 generalises to all graphs: first promote all inner vertices to sources, and then build a large parallel composition of appropriately permuted and lifted edges. For instance, the leftmost graph also admits the following parsing:

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

This defines a notion of *canonical parsing* we use in §3.16 to retrieve a result analogue to the axiomatisation of all graphs by Courcelle and Engelfriet.

While such a parsing exists for all graphs, it goes through an intermediate graph with many sources. 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 3.9.1, [1, Proposition 4.1]). This syntax for denoting precisely the graphs of treewidth at most k is the starting point for the present thesis.

Our goal is to understand which laws are satisfied by the previous operations. Among 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

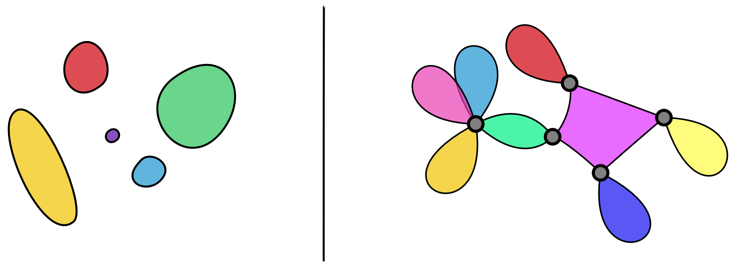
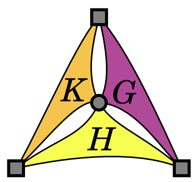


Figure 3: The left and right parts depicts the connected and biconnected components of two graphs, respectively. On the right part, the vertices are the *cutvertices* of the graph, that is the separators reduced to one vertex.

also parse the first graph of Figure 2 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.

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



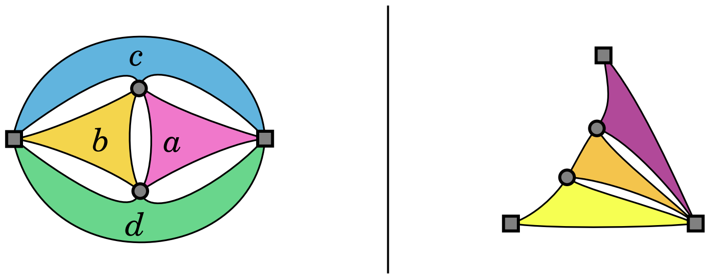
$$\circ(G, H, K) \triangleq \mathsf{f}\big((14)\mathsf{l}G \parallel (24)\mathsf{l}H \parallel (34)\mathsf{l}K\big)$$

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*.

3. Connectivity oriented proofs. To prove the existence of a finite equational axiomatisation of graphs of treewidth at most three, we perform a succession of connectivity decompositions on sourced graphs (Chapter 4). These are reminiscent of the usual connectivity decompositions of graphs into connected and biconnected components (Figure 3).

First, we 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 2.1 actually provided an example of such a decomposition, which is always unique for a given graph. Using a few simple axioms, we show that every term can be rewritten under such a form (Proposition 4.3.1). This makes it possible to focus on full prime graphs in the second step, which is where the main difficulties arise.

4. Forget points and axioms. A property of full prime graphs of bounded treewidth is that they either are *atomic* (i.e., reduced to a permutation of an edge), or have a *forget point*: at least one of their inner vertices can be promoted to 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.



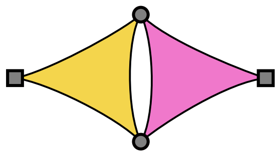
Each of the two inner vertices is a forget point at treewidth at most three, so modulo appropriate permutations of edges, we may parse this graph as $\mathsf{f}(\circ(a,b,c) \parallel d)$ or as $\mathsf{f}(\circ(a,b,d) \parallel c)$. In this case, the two forget points are relatively close, and one of 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.

To do so, we first delimit two classes of vertices which we call *clique points* and *anchors*. The latter can be thought of as a generalisation of cutvertices. When a full prime graph has a clique point or an anchor, we show that we can use it as a universal agreement point.

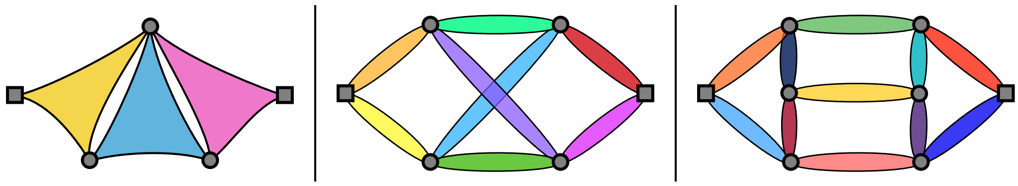
5. Hard graphs. Unfortunately, there are graphs without clique points and anchors. We call them *hard*. The rest of the proof consists in a structural analysis of hard graphs of treewidth at most three.

We show that they admit *separation pairs*: every hard graph has the following shape, and every parsing can be rewritten as a double forget on some of these separation pairs (Proposition 4.26.2):



However, separation pairs are not unique. In order to finish the proof, we analyse how separation pairs relate to each other. We prove the following:

Theorem 5.1. *Every hard graph of treewidth at most three has one of the following three shapes:*



The separation pairs appearing in these shapes are used as agreement point to finish the proof of completeness.

6. (In)completeness results. In this thesis, we prove a series of nine completeness and incompleteness results for different axioms and graph sets.

First we provide a finite equational axiomatisation for graphs of treewidth at most one (§2.15), that is for forests, but with a syntax slightly different from the one presented above (§2.6), and using seven axioms (§2.13).

For the syntax presented above, our results are detailed in Figure 4, for several axiom sets with the following relations:

$$\mathrm{E} \subsetneq \mathrm{C} \subsetneq \mathrm{A} \subsetneq \mathrm{F}_3.$$

Each axiom set can be considered on graphs of treewidth at most k for a non-negative integer k (bounding the arity at $k + 1$), or on the set of all graphs (with unbounded arity).

Each axiom set is obtained from the previous one through the addition of one or two axioms, and E contains thirteen axiom schemes (each being instanciated at different arities).

GRAPH STRUCTURES

7. Treewidth at most two. On top of the theorem above we prove several structural results, including the following well-known structural characterisation of graphs of treewidth at most two:

Axiom sets	Completeness	Incompleteness
E	Treewidth at most 0	Treewidth at most $k \geq 1$
C	Treewidth at most 1 The set of all graphs	Treewidth at most $k \geq 2$
A	Treewidth at most 2	Treewidth at most $k \geq 3$
F_3	Treewidth at most 3	

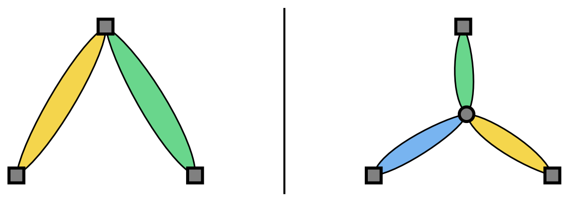
Figure 4: Completeness and incompleteness results for different axiom and graph sets.

Theorem 7.1. *A graph has treewidth at most two if and only if it does not contain K_4 as a minor.*

8. Excluding a complete graph as a sourced minor. The proof of Theorem 5.1 uses a characterisation of graphs of arity three not containing the triangle K_3 as a sourced minor. *Sourced minors* are related to rooted minors (§5.6). Rooted minors are one of the main tools in the graph minor theory of Robertson and Seymour [32]. To obtain a source minor we can, as usual, delete vertices and edges, and contract edges; source keep their status, so to obtain K_3 as a sourced minor in a graph of arity three, edges incident to sources cannot be contracted.

We prove the following which was originally proved in [29] in the context of rooted minors:

Proposition 8.1. *Every graph of arity three has either K_3 as a sourced minor, or one of the following two shapes:*



Moreover, these two shapes do not have K_3 as a sourced minor.

We attempted to generalise our finite equational axiomatisation of graphs of treewidth at most three to graphs of treewidth at most four. Following the structure of our proof, we generalise Proposition 8.1 to graphs excluding the tetrahedron K_4 as a sourced minor in the context of treewidth at most four. This result depends on webs.

A *web* is built by filling with cliques the inner faces of a *near triangulation of the plane*—a simple planar graph whose faces are all bounded by triangles,

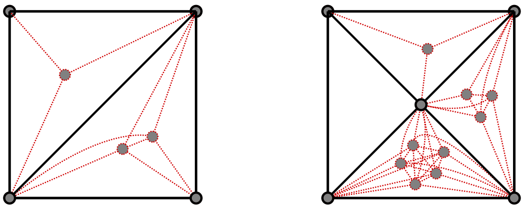


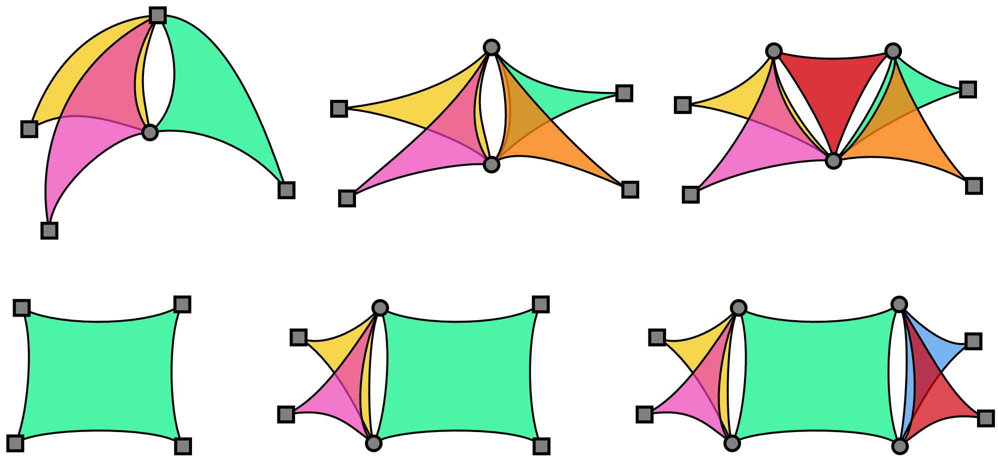
Figure 5: In each of these two webs, the black vertices and edges form a near triangulation of the plane, while the red dotted vertices and edges form cliques filling the triangles of the triangulations.

except for its outer face which is bounded by a cycle—whose triangles are all faces and outer face a cycle of length four).

Examples are given in Figure 5.

Our result for graphs of treewidth at most four is based on the following theorem for general graphs:

Theorem 8.2 ([29]). *Every arity four graph has either K_4 as a sourced minor, or one of the following six shapes:*

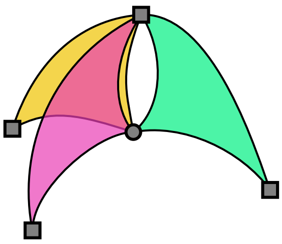


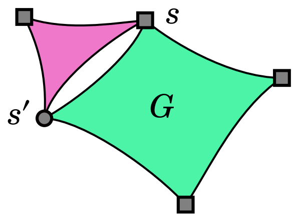
where the edges of arity four are substituted with webs.

Moreover, none of these shapes has K_4 as a sourced minor.

The shapes in Theorem 8.2 do not have treewidth at most four. Using the tools we introduced to handle forget points we refine this theorem in the case of graphs of treewidth at most four. Its proof relies on the following class of graphs.

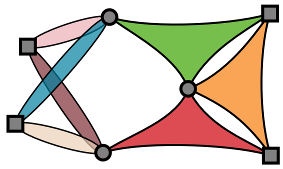
A graph is called *star-like* if it has one of the following shape:





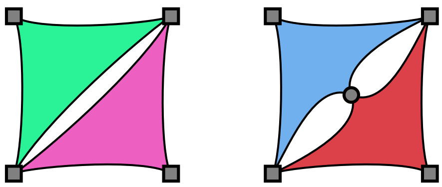
where, for the graph on the right, G is star-like and s and s' are adjacent sources in G.

Theorem 8.3. *Every graph of arity four and treewidth at most four either has K_4 as a sourced minor, or is a spanning subgraph of a web, or has a star-like shape, or has the following shape:*



9. Webs of treewidth at most four. We prove the following to refine subgraphs of webs of treewidth at most four. A *sourced web* is a web in which the four vertices of the outer face are promoted to sources.

Proposition 9.1. *Spanning subgraphs of sourced webs of treewidth at most four have one of the following two shapes:*



where the edges of arity four are again spanning subgraphs of sourced webs of treewidth at most four.

THE TWO DISJOINT PATHS THEOREM AND PROBLEM

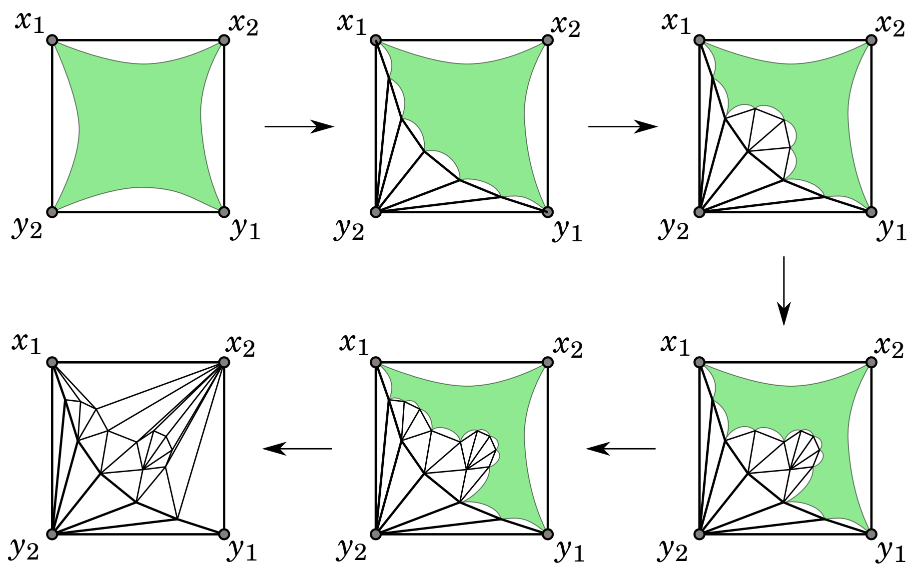
10. Proving the two paths theorem. Webs appear in the statement of Theorem 8.2. Its proof relies on a structural characterisation of graphs without disjoint paths between two disjoint pairs of vertices, often called the *two paths theorem*. It was first proved in [43, 38] and later in [34, (2.4)]. This last proof is presented on its own in [24, Appendix A]. Historically, weaker statements nonetheless close to the whole theorem were proved in [22, 39]. We give a new proof of the two paths theorem, avoiding the use of Kuratowski's or Menger's theorem, and instead relying on an operational characterisation of webs. This technique is similar to that of [34, 24] where the authors nonetheless employ Menger's theorem.

Here, we consider simple graphs. For $k \geq 1$ an integer, in a graph $G = (V, E)$ with a tuple $T = (s_1, s_2, t_1, t_2)$ of four vertices, a *linkage* of T is a set of two disjoint paths, from s_1 to t_1, and s_2 to t_2, respectively. A graph containing at least four vertices and in which for any such tuple there is a corresponding linkage is called 2*-linked*.

Theorem 10.1 (Two paths theorem, [43, 38])**.** *The webs are exactly the edge-maximal graphs which are not* 2*-linked.*

Our proof proceeds as follows for the difficult part of the theorem, that is proving that every edge-maximal graph which is not 2-linked is a web.

We start with G an edge-maximal graph which is not 2-linked: in G, there exists four vertices x_1, y_1, x_2, y_2 such that G does not contain a (x_1, x_2, y_1, y_2)-linkage, and any addition of an edge to G makes it 2-linked. At first, we only know that G contains the cycle $x_1x_2y_1y_2x_1$. Indeed, edges in this cycle cannot be used in a (x_1, x_2, y_1, y_2)-linkage. Starting from y_2 we recursively work our way towards x_2 along some paths. On the way, we reveal the web structure of G. The process is illustrated as follows:



These successive steps are done recursively by generalising the notion of webs. We introduce *k-webs*, which are webs for which the outer cycle—called the *frame*—of the corresponding near-triangulation of the plane is allowed to have any length.

In a graph, a cycle $x_1x_2 \ldots x_kx_1$ is called *crossed* when there are two vertex-disjoint paths from x_i to x_j and x_r to x_s with $i < r < j < s$; it is *crossless* otherwise. The graph G is *maximally C-crossless* when C is crossless and in any edge addition to G, C crossed.

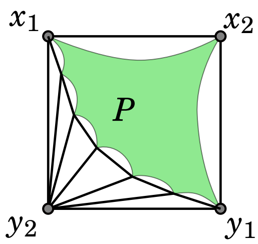
We prove the following generalisation of the two paths theorem:

Theorem 10.2. *Let G be a graph with C a cycle. The following are equivalent:*

- *G is a web with frame C,*
- *G is maximally C-crossless.*

This generalisation first appeared in [34] in the form of *societies*. A *society* is a graph given with a permutation of a subset of its vertices. The latter being a cycle on the graph's vertices (not necessarily a subgraph), we prefer to use the most standard terminology.

Thanks to this generalisation, we can conclude after the first step of the proof sketch above:



This graph is a parallel composition of a web (the bottom left part) with frame $x_1Py_1y_2x_1$ (where x_1Py_1 denotes the subpath of P from x_1 to y_1) and of the green coloured area which is bounded by the cycle $x_1Py_1x_2x_1$. This parallel composition does not identify sources but subpaths of frames. It is called *web composition* when applied on induced paths and satisfies the following:

Proposition 10.3. *Webs are stable under web composition.*

More precisely, the web composition along P and P' of two webs G and H, with respective frames P_GP and P_HP', is a web with frame $P_G\overline{P_H}$.

By induction hypothesis, the green part above is a web, and this proposition allows us to conclude that G is a web.

11. Solving the two disjoint paths problem. The algorithmic problem of finding vertex-disjoint paths between fixed pairs of vertices of a given graph has been well-studied. It is NP-complete on directed graphs [16], and polynomial on undirected graphs [38, 39, 43].

Many variations of this problem exist, such as optimisation versions [40], acyclic case [30], etc. In this thesis, we focus on the following variant: given vertices s_1, t_1, s_2, t_2 in a simple undirected graph, find vertex-disjoint paths respectively from s_1 to t_1 and from s_2 to t_2. If $s_1 = s_2$ or $t_1 = t_2$, then this problem is solved by Menger's theorem: say $s_1 = s_2$; the two paths exist if and only if every vertex separator between s_1 and $\{t_1, t_2\}$ has at least two elements. Various methods relating to flow networks exist to solve such a problem. The difficulty arises when $s_1 \neq s_2$ and $t_1 \neq t_2$. The corresponding algorithmic problem is usually called the *two disjoint paths problem*. Menger's theorem would apply to find vertex-disjoint paths between $\{s_1, s_2\}$ and $\{t_1, t_2\}$, but requiring paths from s_1 to t_1 and s_2 to t_2, respectively, makes the problem harder.

Since the seminal works in [38, 39, 43], many efficient algorithms have been introduced: [26, 17, 41, 42, 23] give, respectively, algorithms running in $O(n^2)$, $O(n+m\log n)$, $O(n+m\alpha(m,n))$, $O(m+n\alpha(n,n)$, and $O(n+m)$ where n and m are respectively the vertex and edge number of the graph, and α is the two parameters variant of the inverse of the Ackermann function. All but the last are based on improving the $O(nm)$ algorithm by Shiloach [39]. The latter relies on a sequence of six reductions, three of which involve previous works: instances can be reduced to the 3-connected case (e.g. using the algorithm from [19] to decompose a graph into its triconnected components), and even the 3-connected non-planar case [30], both in $O(n+m)$ time. By Kuratowksi's Theorem the resulting instance contains a topological $K_{3,3}$ or K_5. Another reduction allows one to assume that the instance does not contain a K_5 [47]. To quote Shiloach "We have not written the solution as an explicit, long, and tedious algorithm".

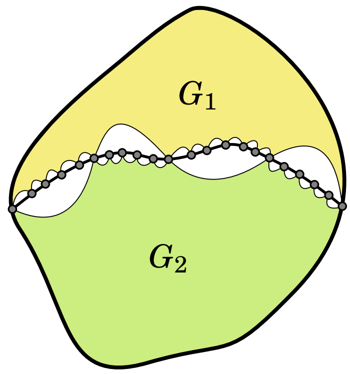
We use the insights given by our proof of the two paths theorem to design a simple $O(nm)$ algorithm for a slight generalisation of the two disjoint paths problem analogous to the generalisation of the two paths theorem described above. Given a tuple T of at least three vertices in a graph G, we call *web completion* of G with regard to T a set F of edges such that $G+F$ is a web with frame T.

Two Disjoint Paths or Web Completion
Input: A graph with a tuple of distinct vertices.
Output: A crossing or a web completion.

Our algorithm departs from the method of Shiloach in that no reductions are required: it is a simple recursive algorithm and, except for a search algorithm (e.g. depth-first or bread-first), it is self-contained. It is given in Chapter 6 in the form of two procedures consisting of ten lines of pseudo-code each.

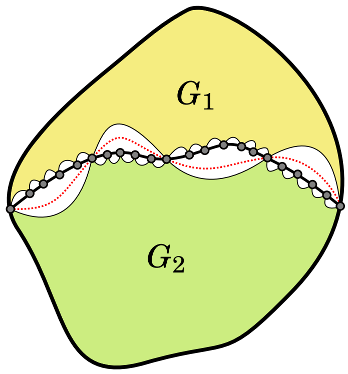
An algorithm following a similar method as ours can be extracted from the proof of the two paths theorem given in [34, 24]. It requires first a reduction to the 3-connected case, but also to find all 3-separators of the instance.

The principle of our algorithm, in the decision version of the problem, goes as follows: given a graph G with a tuple of vertices T, if there are no paths in G between non-consecutive vertices of T, then return No—T is not crossed in G. Otherwise, compute a path P between non-consecutive vertices of T. If P does not separate T, then return No. Otherwise, G has the following shape:



Let T_1 and T_2 be the two parts of T which are separated by P, The graphs G_i with tuples T_iP are then used as subinstances for recursive calls.

Applying this process blindly is not sufficient. Indeed the recursive calls may not be safe. More precisely, it is possible that PT_i is crossed in G_i while T is not in G (for some $i \in \{1, 2\}$). To ensure safeness we first compute what we call a *P-completion*. Roughly, this is a set of safe edges on P's vertices we add to G such that, using these edges, P contains a shorter path P' whose associated subinstances are safe. Computing P-completions roughly correspond to adding the red dotted edges in the following picture:



OUTLINE

In Chapter 1 we introduce basic definitions for simple graphs. We discuss the various characterisations of treewidth and the principles behind connectivity-oriented proofs more thoroughly. Chapter 2 introduces the main tools from universal algebra to define grammars of terms, axioms, and equational reasoning. We provide a first finite axiomatisation for graphs of treewidth at most one, that is for forests. In Chapter 3 we introduce the right notion of sourced hypergraphs allowing us to use syntactic tools, such as variable substitution, directly on graphs. We formally introduce the problem of finite equational axiomatisation for graphs of bounded treewidth, and provide an equational axiomatisation for all graphs, using the axiom set C. Chapter 4 consists of the succession of reductions described above (decomposition into full prime components, along clique points and anchors, etc). These results apply for any bound on the treewidth, but are not sufficient to obtain finite equational axiomatisations at every treewidth. We study hard graphs of treewidth three in Chapter 5 and conclude for graphs of treewidth at most three. Chapter 6 contains our proof of the two paths theorem, and our algorithm for the two disjoint paths problem. Finally Chapter 7 presents our attempt towards a finite equational axiomatisation of graphs of treewidth at most four.

A detailed table of contents is provided at the end of the thesis. It might be used in place of an index.

PUBLISHED AND UNPUBLISHED MATERIALS

This thesis is based on two contributions. At the time of writting, they are published in ICALP 2024 [15] and arXiv [21], respectively.

Definitions and results from Chapters 1 and 2 are mostly standard. The main contributions in these chapters are:

- a variant of the standard hyperedge replacement grammar of graph operations [12, Section 2.4.5] with a proof that the generated graphs are precisely graphs of bounded treewidth (§1.16)—which is already included in [15],
- a finite equational axiomatisation of forests (Theorem 2.16.1) for yet another syntax specifically adapted to forests (§2.6).

The results from [15] are contained in Chapters 3, 4, and 5.

Chapter 3 provides a better account of graph substitution, contexts, and shapes than [15] does, as well as a finite equational axiomatisation of all graphs which is not included in [15] (§3.16).

Chapters 4 and 5 contain revised and extended versions of the main contributions of [15]. In particular, results in Chapter 4 are adapted to graphs of treewidth at most k for any non-negative integer k, whereas [15] presented them for treewidth at most three only. Some results sketched in [15] are now fully formalised, such as the existence of equational axiomatisations of graphs of treewidth at most k when $k \leq 2$ (§4.9, §4.14 and §4.20). The presentation of Chapter 4 with a sequence of axiom sets is new, as well as the definition and treatment of clique points. In [15], clique points were a part of anchors. The incompleteness results are also new (§4.10, §4.15, and §4.21). In [15], the structural property of hard graphs of treewidth at most three is not explicitly stated (Theorem 5.8.1), and neither is the link between hard graphs of treewidth at most $k+1$ with minimal excluded minors of treewidth at most k (Proposition 5.10.1).

Chapter 6 and [21] are very similar, with only a few additions: remarks making links with the graphs operations described in §1.16 and with notions presented in Chapter 3.

Finally, the results of Chapter 7 are entirely new.

The author would like to mention the two contributions [20, 5] that were not included in this thesis since they do not fall in its scope.

1

Graphs and bounded treewidth

1.1 Introduction. In the first section of this chapter (§1.2 to §1.10), we review standard definitions, notations, and results from graph theory (isomorphisms, subgraphs, paths and cycles, connected components...). All these can be found in graph theory textbooks, such as [13, 9].

Treewidth is introduced in the second section (§1.11 to §1.16), via several characterisations using k-trees, k-elimination graphs, tree decompositions, or some graph operations. The first three characterisations originate from [2], [33], and [49, 37], respectively.

The third and last section of this chapter (§1.17 to §1.19) reviews the usual decomposition of graphs along their cutvertices and the principles behind connectivity-oriented proofs. We apply the latter with analogues of connected components and cutvertices in Chapter 4.

Except for §1.16 (in which we describe an operational characterisation of treewidth departing from similar results in the literature, such as in [12, Section 2.4.5], in that we use distinct, though similar, graph operations), all results presented in this chapter are standard. We nonetheless provide complete proofs for the sake of completeness, and citations are provided where we are unable to do so.

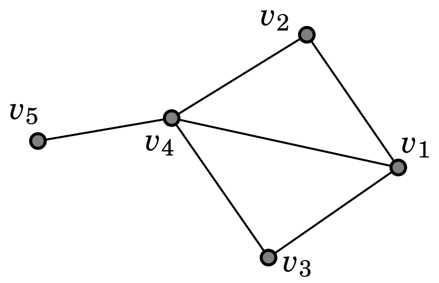


Figure 1.1: A visual representation of the graph with vertex set $\{v_1, v_2, v_3, v_4, v_5\}$ and edge set $\{v_1v_2, v_1v_3, v_1v_4, v_2v_4, v_3v_4, v_4v_5\}$.

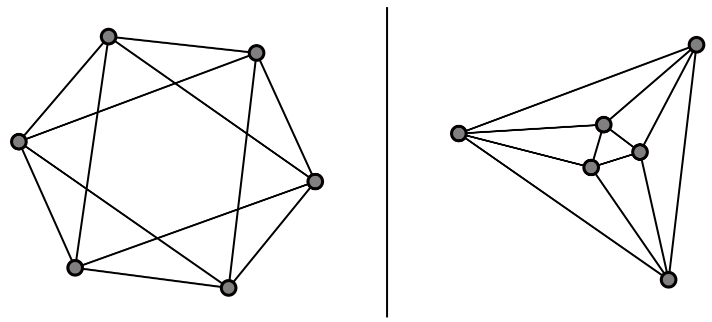


Figure 1.2: Two distinct drawings of the same graph.

We use standard set-theoretic notations: $\{x, y, \dots\}$ for sets, $A \cup B$ for union, $A \cap B$ for intersection, and $A \backslash B$ for set difference. The *powerset* of a set A, that is the set of all subsets of A, is denoted by $\mathcal{P}(A)$ ($\mathcal{P}$ stands for "*powerset*"). Let k be a non-negative integer. The restriction of $\mathcal{P}(A)$ to subsets of size k is denoted by $\mathcal{P}_k(A)$.

We denote the set of non-negative integers by $\mathbb{N}$.

SIMPLE GRAPHS: BASIC DEFINITIONS

1.2 Simple graphs. A *simple graph* is a pair $G = (V, E)$, where V is a finite set of *vertices* and E is a finite set of *edges*, with each edge being an unordered pair of distinct vertices.

Graphs are depicted in the standard manner: vertices are represented by small circles, and edges by lines, as illustrated in Figure 1.1. Graph drawings are not unique; Figure 1.2 provides two distinct representations of a graph.

The *order* of a graph $G = (V, E)$, denoted $|G|$, is the cardinality of V, while the *size* of G, denoted $\|G\|$, is the cardinality of E.

1.3 Basic graphs. The graphs of the form $(V, \emptyset)$ are called *discrete*. The *empty graph*, denoted $\emptyset$, is the unique graph with no vertices, i.e., $(\emptyset, \emptyset)$. Together with the graph $(\{*\}, \emptyset)$, they are referred to as *trivial graphs*.

The graph $K_n = (V, \mathcal{P}_2(V))$, defined on n vertices, is called the *complete graph* or *clique* on n vertices. The graph K_3 is called the *triangle*.

1.4 Vertices and edges. Let $G = (V, E)$ be a graph. Given vertices $u, v \in V$, the edge $\{u, v\}$ is interchangeably denoted uv or vu. We write $V(G)$ and $E(G)$ to refer to the vertex and edge sets of the graph G, respectively. If $e = uv \in E$, then u and v are called *adjacent*, that e is *incident* to u or v, or that u and v are the *endpoints* of e. We also say that v is a *neighbour* of u. The set of all neighbours of u is called the *neighbourhood* of u, and is denoted $N_G(u)$. The *degree* of u is the cardinality of $N_G(u)$. A vertex is called *isolated* if its degree is zero.

In the graph depicted in Figure 1.1, the vertices v_4 and v_5 are adjacent; the former has degree four and the latter degree one.

1.5 Graph operations. Let $G = (V, E)$ be a graph, U a finite set, and $F \subseteq \mathcal{P}_2(V)$.

We write $G + U$ for the graph $(V \cup U, E)$. If $U = \{v\}$ is a singleton, we write $G + u$ and call *vertex addition* the corresponding operation.

We write $G + F$ for the graph $(V, E + F)$. If $F = \{e\}$ is a singleton, we write $G + e$ and call *edge addition* the corresponding operation.

Let $u \in V$ be a vertex and $e \in E$ an edge of G. The *deletion* of u from G is defined as the graph

$$G - u = (V \setminus \{u\}, \{vw \in E \mid v \neq u \wedge w \neq u\}),$$

and the *deletion* of e from G is the graph

$$G - e = (V, E \setminus \{e\}).$$

These operations are referred to as *vertex deletion* and *edge deletion*, respectively.

Let $uv \in \mathcal{P}_2(V)$ be an edge on G's vertices. The *edge contraction* of uv in G is the graph

$$G/uv = (V \setminus \{u\}, \{xy \in E \mid u \neq x \wedge u \neq y\} \cup \{xv \mid xu \in E\}).$$

That is, u is removed from G, and v is made adjacent to all neighbours of u.

Swapping the roles of u and v yields the same graph.

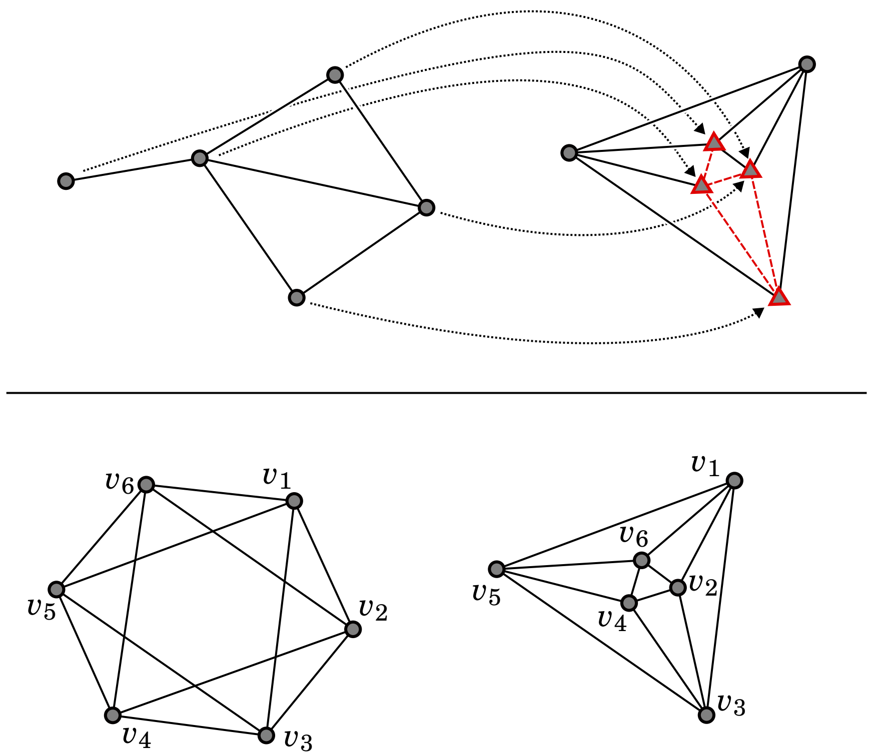


Figure 1.3: The top part depicts a graph homomorphism, while the bottom part illustrates a graph isomorphism. Dotted arrows represent the vertex mapping associated with the homomorphism. The red triangular vertices and red dashed edges are the ones reached by the morphism. Concerning the isomorphism, the bijection is given by the mapping $v_i \mapsto v_i$ ($i = 1, \dots, 6$).

1.6 Graph morphisms. Let G and H be two graphs and $\varphi\colon V(G) \to V(H)$ a function.

The mapping φ is called a *graph homomorphism* from G to H if for every edge $uv \in E(G)$, the image $\varphi(u)\varphi(v)$ belongs to $E(H)$.

In other words, a graph homomorphism is a vertex map preserving adjacency.

If φ is a bijection and both φ and φ^{-1} are graph homomorphisms, φ is called a *graph isomorphism*, G and H are called *isomorphic*, and we write $G \simeq H$.

Isomorphic graphs differ only in the "labelling" of their vertices.

A graph homomorphism and a graph isomorphism are depicted in Figure 1.3. These mappings relate the graphs shown in Figures 1.1 and 1.2.

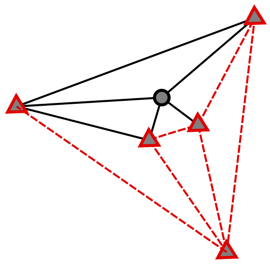


Figure 1.4: The graph in Figure 1.1 as a subgraph of the graph in Figure 1.2. The red triangular vertices and the dashed red edges constitute the subgraph.

1.7 Graph classes, properties, and parameters. A *graph class* is a collection of graphs closed under isomorphism.

In some contexts, graph classes are understood as properties: a class contains a graph if and only if the graph satisfies a certain property. In this sense, a graph class may also be called a *graph property*. Given a graph property φ and a graph G, we write $\varphi(G)$ to indicate that G satisfies the property φ.

A *graph parameter* is a mapping from the collection of all graphs to $\mathbb{R}$ invariant under isomorphism.

For example, the minimum degree of a graph is a graph parameter, while the property of having minimum degree at most one is a graph property. The graph shown in Figure 1.1 satisfies this property, whereas the graph in Figure 1.2 does not.

1.8 Subgraphs. Given two graphs G and H, we say that H is a *subgraph* of G if there exists an injective graph homomorphism from H to G.

An example is provided in Figure 1.4.

Proposition 1.8.1. *A graph H is a subgraph of a graph G if and only if it can be obtained from G by a sequence of vertex and edge deletions.*

Proof. Let $\varphi\colon V(H) \to V(G)$ be an injective graph homomorphism from H to G. Let $V' \subseteq V(G)$ be the image of $V(H)$ under φ, and $E' \subseteq E(G)$ be

$$\{\varphi(u)\varphi(v) \in E(G) \mid uv \in E(H)\}.$$

Then H is isomorphic to the graph (V', E') and (V', E') is obtained from G by successively deleting the edges in $E(G)\setminus E'$ and the vertices in $V(G)\setminus V'$.

Conversely, let H be a graph obtained from G by successive vertex and edge deletions. The mapping from $V(H)$ to $V(G)$ defined by identity is an injective graph homomorphism. □

Let G be a graph. Given a subset $V' \subseteq V(G)$ of vertices, the subgraph of G *induced* by V' is the graph

$$G[V'] = (V', \{uv \in E(G) \mid u, v \in V'\}) .$$

According to Proposition 1.8.1, induced subgraphs are exactly the subgraphs obtained via vertex deletions only, without edge deletions.

A subgraph H of G with $V(H) = V(G)$ and $E(H) \subseteq E(G)$ is called a *spanning subgraph* of G.

1.9 Paths and cycles. A *walk* in a graph G is a sequence of vertices $W = x_0 \dots x_n$ such that for each consecutive pair of vertices x_i, x_{i+1} on W $(0 \leq i \leq n-1)$, $x_i x_{i+1}$ is an edge of G. The vertices x_0 and x_n are referred to as the *endpoints* of the path. The integer n is the *length* of W, which we denote by $|W|$. A *path* is a walk in which vertices do not repeat.

Two walks are *disjoint* if they share no common vertices. They are *inner disjoint* if they have no vertices in common except possibly for their endpoints.

Given a walk $W = x_1 \dots x_n$ as well as two integers $1 \leq i \leq j \leq n$, we write $x_i W x_j$ for the *subwalk* $x_i x_{i+1} \dots x_j$ $(0 \leq i \leq j \leq n)$. If $i = 0$ or $j = n$, we simplify the notation to Wx_j and $x_i W$, respectively. The *reversal* of W is the walk $\overline{W} = x_n \dots x_1$ and the *interior* of W is its subwalk $\mathring{W} = x_2 \dots x_{n-1}$. An *induced path* is a path that is induced as a subgraph.

Two vertices u and v in a graph G are *connected* if G contains a walk W with endpoints u and v. In this case, we say that W *connects* u and v in G. A graph G is called *connected* if any two of its vertices are connected. By convention, the empty graph is considered disconnected.

Given two sets U, V of vertices, a path $P = x_1 \dots x_n$ is a *U-V path* if $x_1 \in U$, $x_n \in V$, and the interior of P avoids both U and V. When $U = V$ we simply say that P is a *U-path*. Let W be a set of vertices. The sets U and V are called *W-connected* when there exists a U-V path whose interior avoids W.

Given a walk $W = x_0 \dots x_n$, if a vertex x appears twice in W, say as x_i and x_j with $i < j$, then the walk $W' = x_0 \dots x_i x_{j+1} \dots x_n$ is shorter in length than W but with the same endpoints. Hence x can be used to shortcut the walk W. Repeated applications of this process yields a paths with the same endpoints as W.

Next, we introduce several notations for reasoning with walks and paths.

Given a graph G and walks $P = x_0 \dots x_n$ and $Q = y_0 \dots y_m$, if $x_i = y_j$ for some non-negative integers i and j, Px_iQ denotes the walk $x_0 \dots x_i y_{j+1} \dots y_m$. This notation may be applied iteratively, e.g., $PxQyR$. When the last vertex x of P coincides with the first vertex of Q (i.e., $x_n = y_0$), we write PQ instead of PxQ.

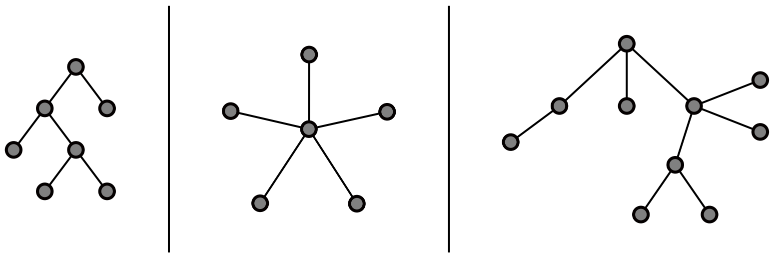


Figure 1.5: Examples of trees.

A *cycle of length n* (with $n \geq 3$) is a walk in which the only repeated vertices are the endpoints.

When two paths P and Q are disjoint except for their first and last vertices which are equal, PQ is a cycle.

1.10 Connected components. The *disjoint union* $G \uplus H$ of two graphs G and H is the graph whose vertex set is the disjoint union of $V(G)$ and $V(H)$, and whose edge set is the disjoint union of $E(G)$ and $E(H)$.

Given a graph G we can consider the equivalence relation R between its vertices defined by connectivity: $(u, v) \in R$ if and only if u and v are connected by some path in G. The associated equivalence classes of R define maximally connected subsets of vertices and the associated induced subgraphs are called the *connected components* of G.

Proposition 1.10.1. *A graph is the disjoint union of its connected components.*

Write C_G for the set of connected components of a graph G and let φ be a graph isomorphism from G to a graph H. We have $\varphi(C_G) = C_H$. For this reason, the decomposition of a graph into its connected components is called *canonical*.

Another graph connectivity decomposition is presented in the end of this chapter.

TREES AND TREEWIDTH

1.11 Trees. An acyclic graph is called a *forest*. If it is additionally connected, it is called a *tree*.

By Proposition 1.10.1 forests are precisely disjoint unions of trees. Examples of trees are shown in Figure 1.5.

Trees and forests are among the most studied classes of graphs. They admit various characterisations, such as the following one.

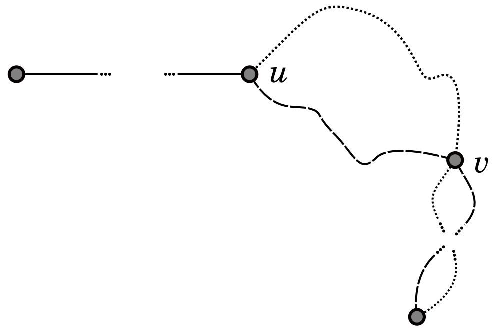


Figure 1.6: Shapes of distinct paths P and Q sharing the same endpoints in a graph. The regular lines represent $Pu = Qu$, the dotted and dashed lines represent uP and uQ. The walk $uPv\overline{Q}u$ is a cycle.

Proposition 1.11.1. *A graph G is a forest if and only if in G, there exists at most one path between every pair of vertices.*

In particular, a graph G is a tree if and only if in G, there exists exactly one path between every pair of vertices.

Proof. If a graph contains a cycle, then the cycle provides two distinct paths between any pair of vertices on the cycle.

Conversely, if a graph contains two distinct (not necessarily inner disjoint) paths P and Q sharing their endpoints, then a cycle is constructed as follows from P and Q (see Figure 1.6).

Let u be the last vertex of P and Q such that $Pu = Qu$. Since the paths P and Q are distinct, u cannot be their last vertex. Furthermore, P and Q share their endpoints. Hence their last vertex is common to the two paths and appears after u. Let v be the first vertex after u on P also appearing on Q. The walk $uPv\overline{Q}u$ is a cycle. □

The rest of the current section is devoted to introducing graphs of bounded treewidth and their most well-known characterisations, drawing parallels with characterisations of trees or forests.

1.12 Treewidth via k-trees. Often, treewidth is understood as a graph parameter expressing how close a graph is to being a tree. We chose to introduce the treewidth of graphs via k-trees in the hope of clarifying this intuition. We first introduce the corresponding characterisation for trees.

In a graph, a vertex of degree at most one is called a *leaf*. Given a graph G, and $u \in V(G)$, the *u-leaf addition to G* is the graph obtained by adding a new fresh vertex v to G as well as the edge uv.

Proposition 1.12.1. *The closure of the trivial graphs under successive leaf additions is exactly the set of trees.*

Proof. We prove that following two items:

1. A leaf addition to a tree is a tree.
2. Every non-trivial tree contains a leaf.

1. Let G be a leaf addition to a tree T. We write u for the added leaf. For the sake of contradiction, let C be a cycle in G. Every vertex of C has two neighbours on C. Hence C cannot contain u, and C is a cycle in T, a contradiction.

2. For the sake of contradiction, let T be a non-trivial tree not containing a leaf. Since T is non-trivial, every vertex of T has degree at least one. Furthermore, as T does not contain a leaf, every vertex of T has degree at least two.

Let $P = x_1 \dots x_n$ be a path in T of maximal length. As the degree of x_n is at least two, let y be a neighbour of x_n distinct from x_{n-1}. By maximality of P, the walk $x_1 \dots x_n y$ is not a path. Hence y is a vertex x_i of P for an integer $i \leq n-2$, and yPx_ny is a cycle in T. □

We now define k-trees by generalising the definitions of leaves and leaf additions.

In a graph, we call *k-leaf* a vertex of degree k whose neighbours induce a clique. Given a graph G with a clique K of order k, *k-leaf addition to G* is the graph obtained by adding a fresh vertex v to G and every edge uv for $u \in V(K)$. A *k-tree* is a graph obtained, starting with a complete graph on k vertices, by successive k-leaf additions.

The complete graph K_{k+1} is obtained by a k-leaf addition of K_k. Hence, it is a k-tree.

Examples of 3-trees akin to the trees of Figure 1.5 are displayed in Figure 1.7.

Definition 1.12.2. The *treewidth* of a graph is the least integer k such that it is a subgraph of a k-tree.

A graph class has *bounded treewidth* if there exists a non-negative integer k such that all its graphs have treewidth at most k.

Any graph G is a subgraph of $K_{|G|}$. Hence, the treewidth of a graph G is bounded by its order $|G|$.

Leaf additions coincide with 1-leaf additions. Hence, as a direct consequence of Proposition 1.12.1, we have:

Corollary 1.12.3. *A graph is a forest if and only if it has treewidth at most one.*

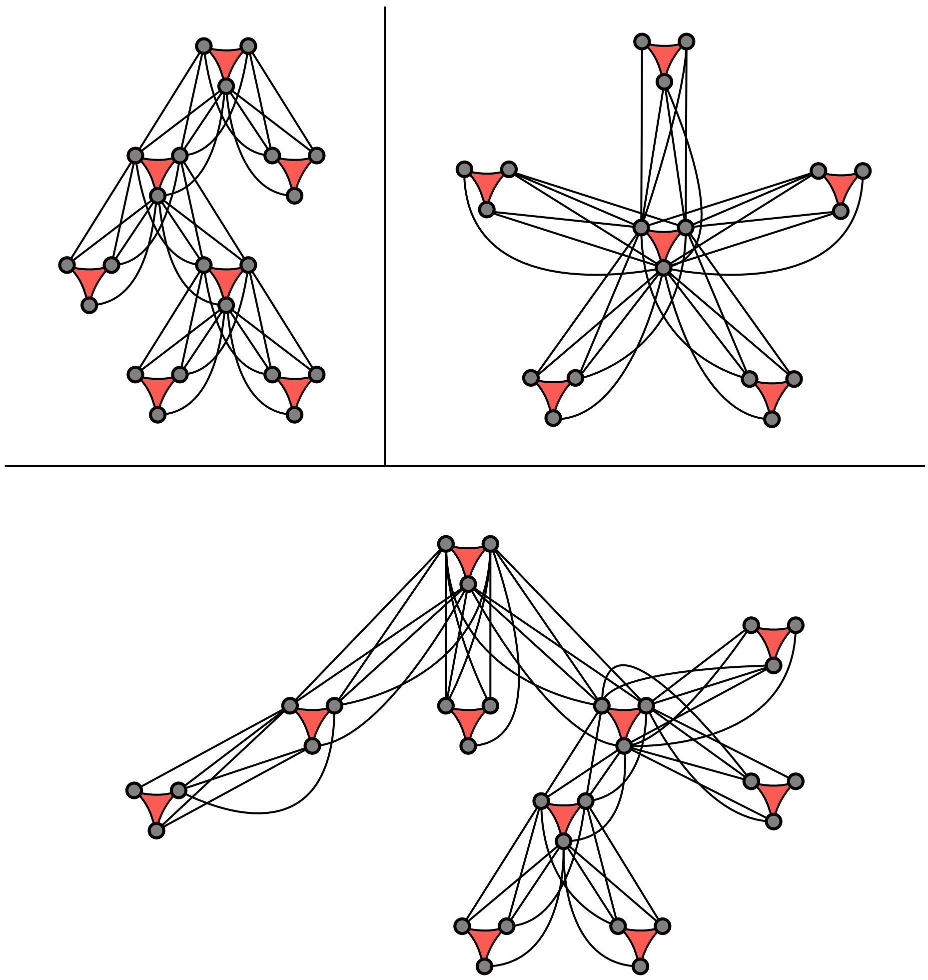


Figure 1.7: Examples of 3-trees. Red triangles are cliques of size three representing the vertices in the trees of Figure 1.5: k-trees can be understood as “thickened” versions of usual trees.

1.13 Treewidth via k-elimination graphs. In the previous paragraph, instead of considering k-leaf additions for a fixed k, one can also allow l-leaf additions whenever $l \leq k$.

The graphs obtained from cliques of order at most k via successive l-leaf additions ($0 \leq l \leq k$) are called *k-elimination graphs*.

The standard definition of elimination graphs follows the inverse process: the k-elimination graphs are exactly the graphs that can be reduced to the empty graph by repeated deletions of l-leaves where l ranges in $\{0, \dots, k\}$.

Clearly, k-elimination graphs are all subgraphs of k-trees and k-trees are all k-elimination graphs.

Proposition 1.13.1. *A graph has treewidth at most k if and only if it is a subgraph of a k-elimination graph.*

The justification behind the introduction of k-elimination graphs is to simplify some proofs.

1.14 Treewidth via tree decompositions. The "tree-like structure" of k-trees is quite clear visually as shown in Figure 1.7. Still, formally, it is difficult to use properties of trees on k-trees, and, by extension, on graphs of bounded treewidth.

Tree decompositions offer an explicit tree structure encoding the treewidth of a graph. In the literature, treewidth is usually introduced using tree decompositions.

Definition 1.14.1. A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (B, F)$ whose vertices are called *bags*, along with a map $\varphi\colon B \to \mathcal{P}(V)$ associating a subset of V to each bag of T, such that:

- $V = \bigcup_{b \in B} \varphi(b)$;
- for each edge xy of G, there exists a bag $b \in B$ with $x, y \in \varphi(b)$;
- for every vertex $x \in V$, if two bags $b, b' \in B$ both contain x ($x \in \varphi(b) \cap \varphi(b')$), then x is contained in every bag along the unique path in T from b to b'.

The *width* of a tree decomposition is the maximum cardinality of its bags minus one.

Figure 1.8 gives visual representations of tree decompositions for the 3-trees of Figure 1.7.

To prove the treewidth of a graph is the minimum width of its tree decompositions, we first prove a lemma on the behaviour of cliques in tree decompositions.

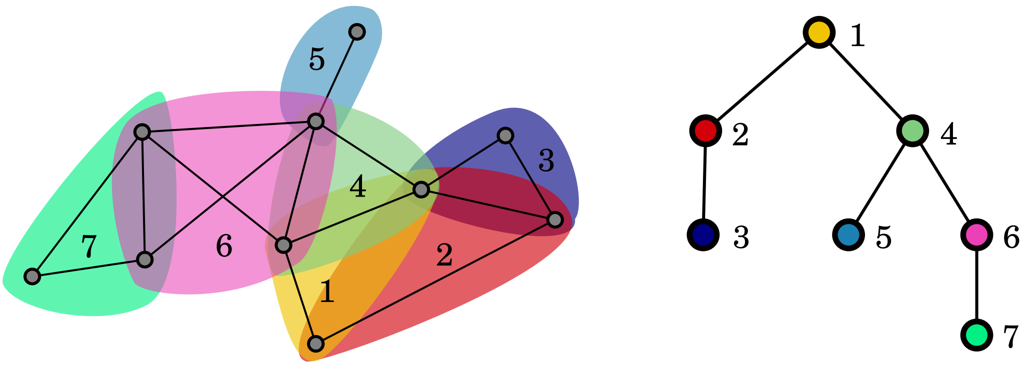


Figure 1.8: On the right part, the tree of a tree decomposition of the graph on the left part. Bags are represented by the coloured areas.

Lemma 1.14.2. *Let G be a graph, K a clique in G, and T a tree decomposition of G. There exists a bag of T containing K.*

Proof. The proof proceeds by induction on $|K|$. If $|K| \leq 2$, then the statement holds by definition of tree decompositions.

Otherwise, $|K| \geq 3$. Let $u, v, w \in K$ be distinct vertices.

By induction hypothesis, there exists three bags b_u, b_v, b_w of T containing $K\backslash u$, $K\backslash v$ and $K\backslash w$, respectively.

By Proposition 1.11.1, consider the unique paths P, Q, and R in T from b_u to b_w, from b_v to b_w, and from b_u to b_v, respectively.

By definition of tree decomposition, all bags on the paths P, Q, and R contain $K\backslash\{u, w\}$, $K\backslash\{v, w\}$, and $K\backslash\{u, v\}$, respectively.

We prove that there exists a bag b common to P, Q, and R.

We distinguish two cases, depending on whether b_w is on R or not. See Figure 1.9 for an illustration.

If b_w appears on R then b_w is common to the three paths P, Q, and R. In this case, we let $b \triangleq b_w$.

Otherwise R does not contain b_w.

Let S be a path obtained by shortcutting the walk PQ, preserving its endpoints. The paths S and R share their endpoints. By Proposition 1.11.1, they are equal. Hence S does not contain b_w. As PQ contains b_w and S is defined by shortcutting PQ, there exists a bag b common to P, Q, and S, such that b_w is contained in $bPQb$.

The bag b appears on the three paths P, Q, and R: by definition of tree decomposition, it contains $(K\backslash\{u, w\}) \cup (K\backslash\{v, w\}) \cup (K\backslash\{u, v\}) = K$. □

Theorem 1.14.3. *A graph has treewidth at most k if and only if it has a tree decomposition of width at most k.*

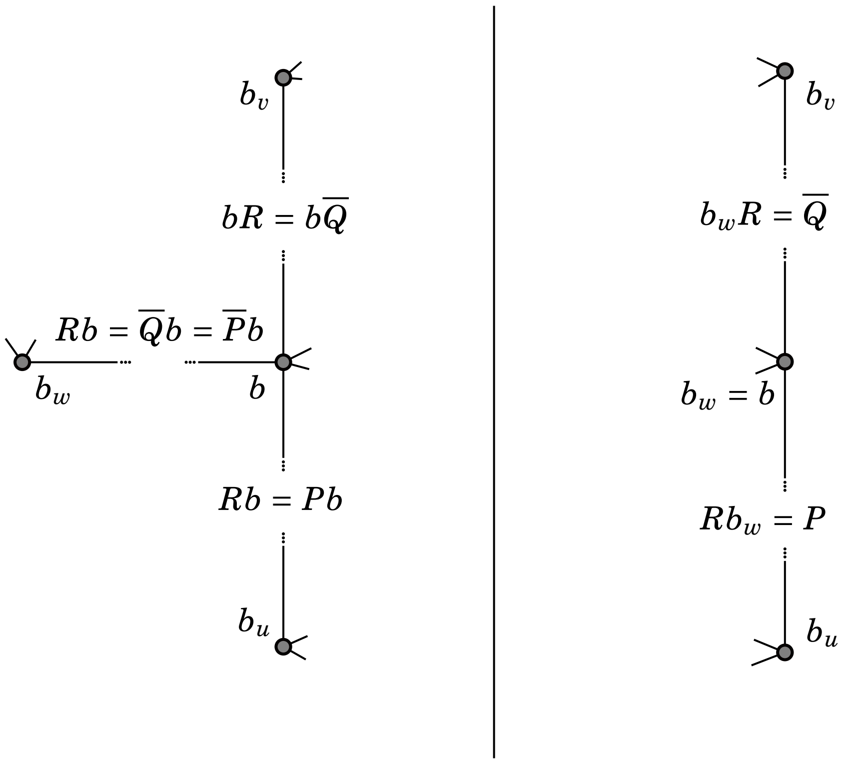


Figure 1.9: The two possible configurations for the relative shapes of the paths PQ and R in the proof of Lemma 1.14.2.

Proof. The two implications are illustrated in Figures 1.11 and 1.10.

Let G be a graph.

Assume a tree decomposition $T = (B, F)$ of G of width at most k. Consider the graph H whose vertices are those of G, two vertices being adjacent in H if and only if they belong to a common bag of T. By the second condition of Definition 1.14.1 and by Proposition 1.8.1, G is a subgraph of H. We prove G has treewidth at most k by proving H is a k-elimination graph.

Write ϕ for the function associating the subsets of $V(G)$ to the bags of B.

We reason by induction on the lexicographic product $(|G|, |T|)$ of G's order with the number of bags in T.

Assume T is reduced to one bag denoted by b. By the first condition of Definition 1.14.1, $V(G) = \phi(b)$. Hence $H \simeq K_{|G|}$. As T has width at most k, we have $|\phi(b)| \leq k + 1$. Hence H is a complete graph with at most $k + 1$ vertices. So H is a k-elimination graph.

Otherwise, T contains at least two bags and by Proposition 1.12.1 there is at least one leaf b in T.

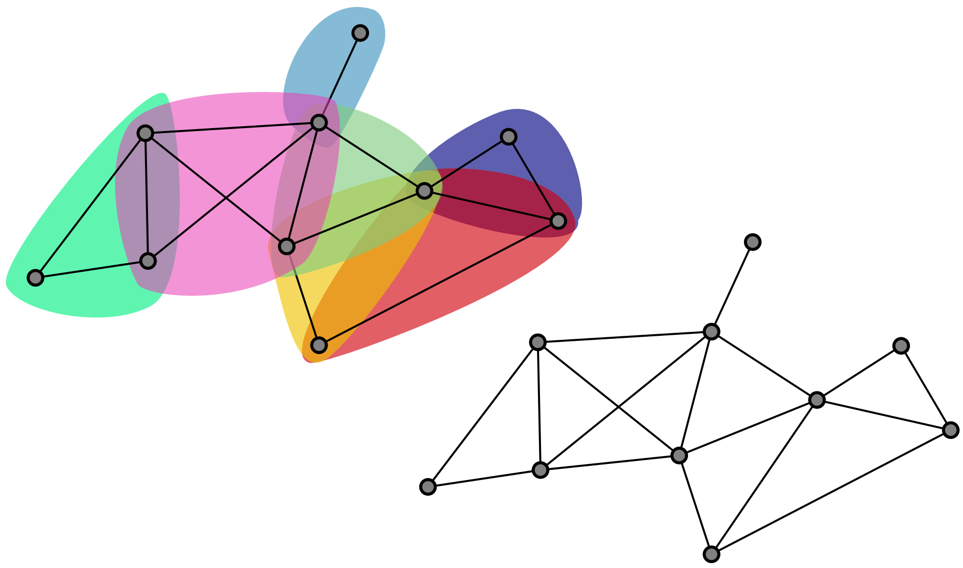


Figure 1.10: On the top part, the graph from Figure 1.9 with the bag of its tree decomposition. On the bottom part, the 3-elimination tree obtained by turning the bags into cliques.

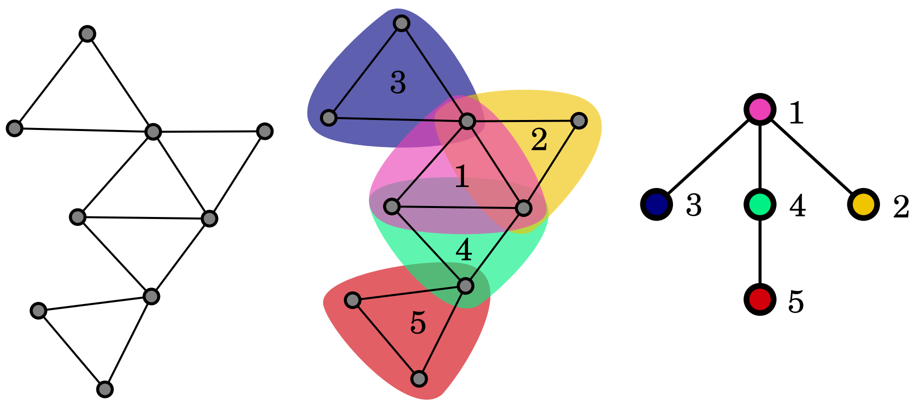


Figure 1.11: On the left part, a 2-elimination tree. On the right part, a tree decomposition of the 2-tree of width at most two whose bags are the maximal cliques of the 2-elimination tree.

We distinguish two cases, depending on whether each vertex of G appears in a bag of $T - b$ or not.

- Assume each vertex of G appears in a bag of $T - b$.

  We prove that $T - b$ is a tree decomposition of G, allowing us to conclude by induction hypothesis as $|T - b| < |T|$.

  The first and third conditions of Definition 1.14.1 are directly inherited from T.

  We prove the second condition of Definition 1.14.1, namely, that for every edge uv of G, there exists a bag of $T - b$ containing both u and v.

  Let b_{uv} be a bag of T containing both u and v. If $b \neq b_{uv}$ then b_{uv} is a bag of $T - b$ containing both u and v. Otherwise, $b = b_{uv}$. Let b' be the only neighbour of b in T and b_u be a bag of $T - b$ containing u. As T is a tree decomposition of G, the third condition of Definition 1.14.1 ensures that u is contained in all bags along the unique path in T from b to b_u. As b is a leaf, this path contains b'. So b' contains u. By symmetry b' also contains v. Hence in $T - b$ there is a bag containing both u and v. So $T - b$ is a tree decomposition of G.

- Otherwise, some vertex $x \in \phi(b)$ appears in b and in no other bag of T.

  Removing x from b provides a tree decomposition of $G - x$. By induction hypothesis, as $|G - x| = |G| - 1$, $H - x$ is a k-elimination graph and $G - x$ is a subgraph of $H - x$.

  As the only bag of T containing x is b, by the second condition of Definition 1.14.1, all neighbours of x are contained in b. By definition of H, the vertices contained in b form a clique in $H - x$. Hence, H is the $(|\phi(b)| - 1)$-leaf addition of x to $H - x$. As T has width at most k, $|\phi(b)| \leq k + 1$. As furthermore $H - x$ is a k-elimination graph, H is a k-elimination graph containing G as a subgraph.

Assume G is a subgraph of a k-elimination graph H. We prove that G and H share a common tree decomposition of width at most k by induction on the definition of k-elimination graphs.

If H is a clique of order at most k, then consider the trivial tree decomposition containing one bag with all vertices in it. It has width at most k and it is a tree decomposition of G.

Otherwise, with $l \leq k$, assume H is given by a l-leaf addition of a vertex v to the k-elimination graph $H - v$. As G is a subgraph of H, $G - v$ is a subgraph of $H - v$. By induction hypothesis, consider a tree decomposition T of width at most k which is a decomposition of both $G - v$ and $H - v$.

The neighbourhood of v in H is a clique by definition of leaf additions. By Lemma 1.14.2, the neighbourhood of v in H is contained in a bag b of T. Let T' be the tree decomposition obtained from T via a b-leaf addition of a new bag b', adding in b' the vertex v as well as its neighbours $N_H(v)$ from H.

The tree T' is a tree decomposition for both G and H of width at most k. □

1.15 Treewidth via excluded minors. Previous paragraphs introduced characterisations of treewidth in which the tree structure is somewhat explicit, be it via successive k-leaf additions, or via tree decompositions. We now consider structural characterisations.

Structural graph theory studies characterisations of graph classes via finite lists of obstructions. Examples are graph classes avoiding a fixed finite set of graphs as subgraphs.

Denote by S_k the *star* with *k branches* ($k \geq 0$). More precisely, S_k consists of $k+1$ vertices $x, l_1, \ldots, l_k$ and the k edges xl_i ($i = 1, \ldots, k$). The vertex x is called the *center* of S_k.

For any non-negative integer k, the class of graphs with maximum degree at most k coincides with the class of graphs that exclude S_k as a subgraph.

For the class of graphs of treewidth at most k, the structural characterisation uses the "minor" relation instead of the subgraph one.

A graph H is called a *minor* of a graph G if there exists a family $(V_x)_{x\in V(H)}$ of pairwise disjoint subsets of $V(G)$ such that:

- $G[V_x]$ is a connected subgraph of G for all $x \in V(H)$, and
- whenever $xx' \in E(H)$ then $yy' \in E(G)$ for some vertices $y \in V_x$ and $y' \in V_{x'}$.

The family $(V_x)_{x\in V(H)}$ is called a *model* of H in G.

An example of minor is shown in Figure 1.12.

In other words, $(V_x)_{x\in V(H)}$ partitions a subset of $V(G)$, with the adjacency between the V_x containing the adjacency relation on H's vertices. The minor relation is a generalisation of the subgraph relation, analogous to how partitions generalise subsets.

Proposition 1.15.1. *A graph H is a minor of a graph G if and only if it can be obtained from G by successive vertex deletions, edge deletions, and edge contractions.*

Proof. Assume H is a minor of G and let $(V_x)_{x\in V(H)}$ be an associated model of H. Perform vertex deletions vertices in $V(G)\backslash(\cup_{x\in V(H)} V_x)$, edge deletion for each edge uv with $u \in V_x$, $v \in V_y$, $x \neq y$, and $xy \notin E(H)$, and edge contraction for each edge of the induced subgraphs $G[V_x]$ ($x \in V(H)$).

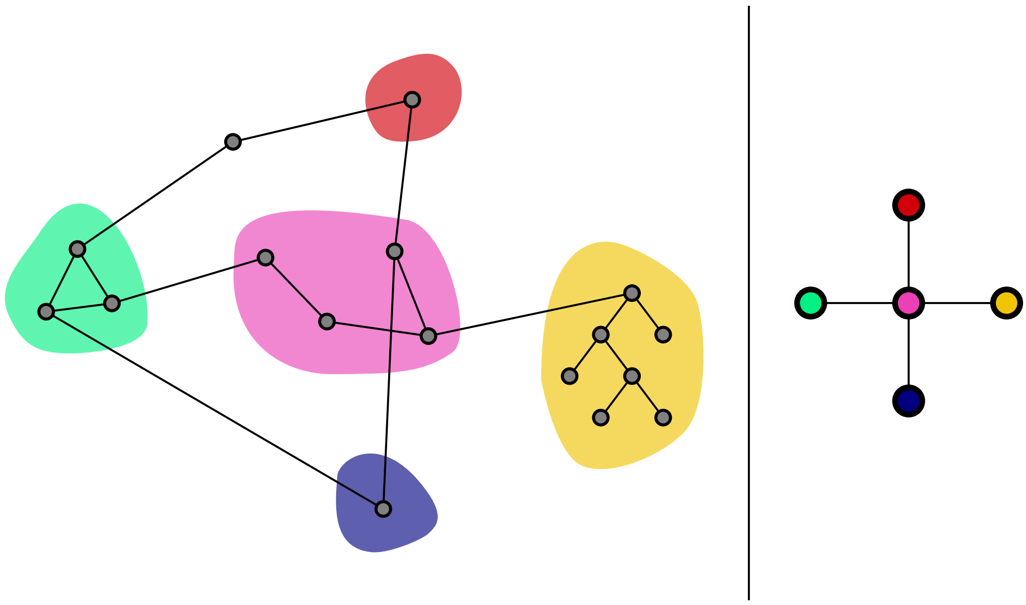


Figure 1.12: The right part depicts a minor of the graph on the left part. Coloured areas represent the vertices which are identified in the minor. Observe that the minor contains a vertex of degree four whereas the graph has maximal degree three.

The resulting graph is isomorphic to H by mapping the vertex resulting from the contraction of edges in $G[V_x]$ to x.

Conversely, we proceed by induction on the sequence of operations performed.

Assume H is a minor of $G - v$ ($v \in V(G)$) or $G - e$ ($e \in E(G)$). The model $(V_x)_{x \in V(H)}$ of H in $G - v$ or $G - e$ is a model of H in G.

Assume H is a minor of G/e for an edge $e \in E(G)$. If the vertex w corresponding to the identification of e's endpoints in G/e does not appear in any set V_x, then $(V_x)_{x \in V(H)}$ is a model of H in G. Otherwise, replace w in its set V_x by e's endpoints, and the corresponding family of subsets provides a model of H in G. □

Proposition 1.15.2. *A graph is a tree if and only if it does not contain K_3 as a minor.*

Proof. We prove that having a triangle K_3 as a minor is exactly as having a cycle.

Let G be a graph containing a cycle $C = x_1 \dots x_n x_1$ of length n. Deleting all vertices and edges in $G - C$ from G and contracting $x_i x_{i+1}$ ($i = 3, \dots, n-1$) provides a triangle as a minor by Proposition 1.15.1.

Conversely, assume that (V_1, V_2, V_3) is a model of a triangle in a graph G. By definition of model, there are vertices $x, x' \in V_1$, $y, y' \in V_2$ and $z, z' \in V_3$ such that $x'y, y'z, z'x \in E(G)$. As the graphs $G[V_i]$ are connected and disjoint, we can assume disjoint paths P, Q, R in G, from x to x', from y to y', and from z to z', respectively. Then $Px'yQy'zRz'x$ is a cycle in G. □

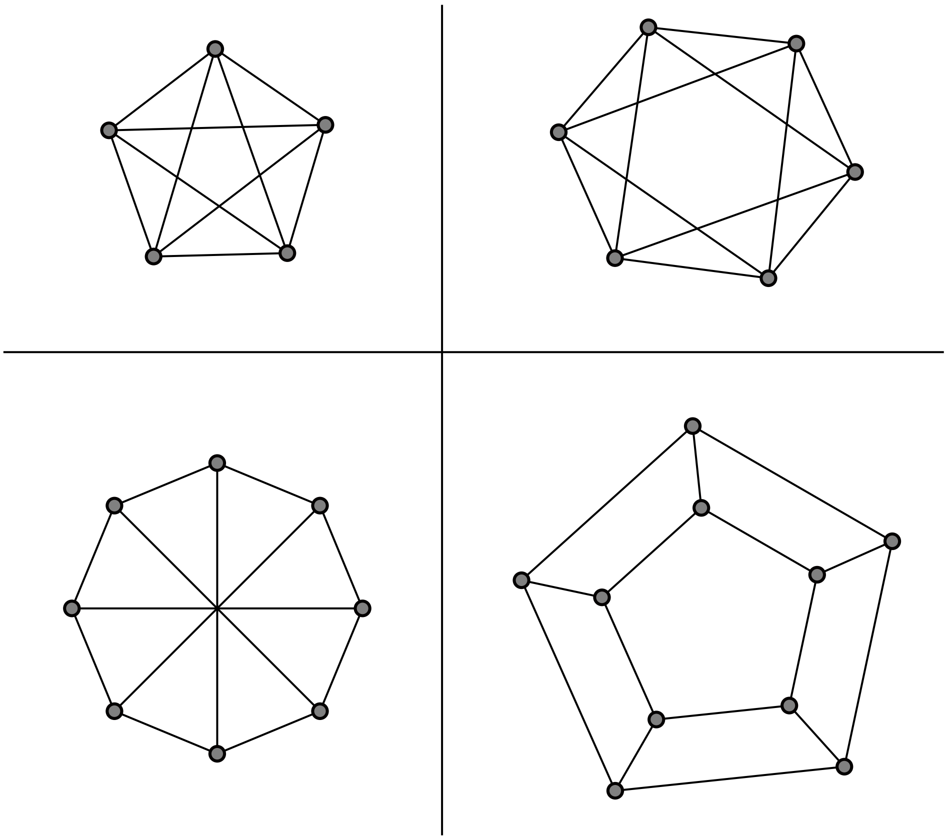
This proposition generalises to graphs of treewidth at most k:

Theorem 1.15.3 ([32])**.** *For every $k \geq 0$ there exists a finite set M_k of graphs such that a graph has treewidth at most k if and only if it contains none of the graphs from M_k as a minor. Moreover, if $G \in M_k$ then every minor of G has treewidth at most k.*

The proof of Theorem 1.15.3 relies on the celebrated Robertson-Seymour theorem, which states, in its simplest form that for any infinite sequence $(G_i)_i$ of simple undirected graphs, there are indices $i < j$ such that G_i is a minor of G_j. This is a very deep result, possibly the most important one in graph theory. Its proof took several decades and the publication of more than twenty research papers. It is nonconstructive, and the explicit lists of minors in Theorem 1.15.3 are unknown for $k \geq 4$. On the other hand, when $k \leq 3$, we have the following:

Theorem 1.15.4. *A graph has treewidth at most two if and only if it does not contain K_4 as a minor.*

Theorem 1.15.5 ([36, 4])**.** *A graph has treewidth at most three if and only if it does not contain one of the following four graphs as minors:*



The associated lists of minors being unknown for $k \geq 4$, the characterisation of graphs of bounded treewidth in Theorem 1.15.3 is of a different flavor than the already seen ones.

We obtain Theorem 1.15.4 in § 5.10 as a byproduct of our study of a finite equational axiomatisation for graphs of treewidth at most three.

In the general case of graphs of treewidth at most k, we know at least one of the forbidden minors:

Proposition 1.15.6. *For every non-negative integer k, the complete graph* K_{k+2} *has treewidth* $k + 1$*. Furthermore all its minors have treewidth* k*.*

Proof. The first part is a direct consequence of Proposition 1.14.2. The second part is proved as follows: the edge contraction or vertex deletion of K_{k+2} yields K_{k+1}. An edge deletion yields a k-leaf addition to K_{k+1}. In any case the resulting graph is a k-tree which has treewidth k. □

1.16 Treewidth via graph operations. A *rooted graph* is a graph in which a particular vertex is identified and called its *root*.

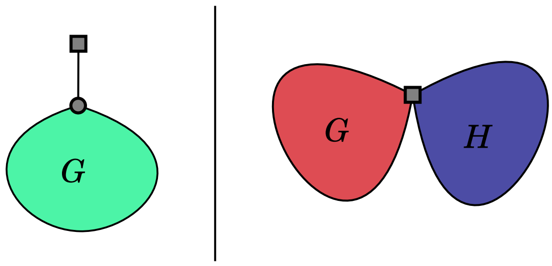


Figure 1.13: On the left and right parts, the shapes of a root lift rl$G$ and of a parallel composition $G \parallel H$, respectively.

Rooted trees are closer to tree representations as datatypes, and are often used in recursive algorithms: they admit inductive characterisations using a finite number of graph operations.

On the other hand, the characterisation of Proposition 1.12.1 via u-leaf additions requires one graph operation per value u can take, amounting to infinitely many graph operations.

Allowing roots changes the graph notion, but defines a clear interface on which to work to generate trees.

Given two rooted graphs G and H with roots r_G and r_H, we call *parallel composition* of G with H the rooted graph $G \parallel H$ obtained by taking their disjoint union, adding the edge $r_G r_H$ in the result, and then contracting $r_G r_H$, the resulting vertex being the root:

$$G \parallel H = (G \uplus H)/r_G r_H.$$

This operation consists in "gluing" the two graphs on their roots.

Given a rooted graph G with root r, the *root lift* of G consists in adding a fresh vertex r' to G, adding the edge rr', forgetting r is the root, and promoting r' as the root.

Examples of parallel composition and root lift are provided in Figure 1.13

The rooted graph consisting solely of its root is called the *trivial rooted graph*.

A way of generating trees from the trivial rooted graph is to recursively apply parallel compositions and root lifts.

To access forests on top of trees, we define two further operations.

The *root lift* of a nonrooted graph G is the graph obtained by adding a fresh isolated vertex r to G, the root of the resulting graph being r; the *forget* of a rooted graph G is the nonrooted graph obtained from G by forgetting it has a root.

The term "root lift" is overloaded and describes an operation on both rooted and nonrooted graphs.

Proposition 1.16.1. *The closure of the set containing the empty graph under root lifts, parallel compositions, and forget operations, is precisely the set of possibly rooted forests.*

Proof. We prove the following two statements of which the proposition is a direct consequence:

1. The root lift of a forest, rooted or not, is a rooted forest. The parallel composition of two rooted forests is a rooted forest. The forget of a rooted forest is a rooted forest.

2. Any non-empty forest is the forget of a rooted forest and any rooted forest can be expressed either as a root lift of a (rooted or not) forest or as a parallel composition of two nontrivial rooted forests.

1. The root lift of a rooted tree is a leaf addition in which the new leaf becomes the root. By Proposition 1.12.1 this is again a rooted tree. A rooted forest being a disjoint union of rooted trees by Proposition 1.10.1, this extends to rooted forests.

The root lift of a nonrooted forest is the exact same graph with one new connected component reduced to the root. Since the latter is a forest, we conclude by Proposition 1.10.1 that the root lift of a nonrooted forest is a rooted forest.

Consider the parallel composition of two rooted forests T and T'. Assume, for sake of contradiction, there exists a cycle C in $T \parallel T'$. If the vertices of C are either all vertices of T or of T', then C is a cycle in T or T', a contradiction. Otherwise, assume there is in C a (non-root) vertex x which is in T but not in T' and a (non-root) vertex y which is in T' but not in T. Consider the largest portion P of C containing x whose vertices are all vertices of T. Because y exists in C, $P \neq C$ and P's endpoints are distinct vertices. Only one endpoint of P can be the root r of $T \parallel T'$, but both are adjacent to a vertex of $T' - r$, a contradiction.

The forget of a rooted forest is a forest.

2. The rooted graph obtained by promoting any vertex as the root in a nonrooted forest is a rooted forest by definition. Hence any nonrooted forest is the forget of a rooted forest.

Assume T is a non-empty rooted forest let r be its root.

If r is isolated, then T is the root lift of the nonrooted forest $T - r$.

Next, consider the case where r is not isolated in T. We distinguish two cases depending on whether there are two connected components in $T - r$ or not.

$T - r$ **has at least two connected components;** let C be one such component: T is the parallel composition of the rooted forests $T[C \cup \{r\}]$ with

$T - C$, and both arguments of the parallel composition are nontrivial as $T - r$ has at least one component distinct from C.

$T - r$ **has exactly one component;** the root being non-isolated, its degree in T is exactly one. Write r' for the unique neighbour of r in T. Then T is the root lift of the rooted tree $T - r$ in which r' is the root. □

The definitions and results we have presented so far are standard in graph theory. Next, we generalise the operational characterisation of forests of Proposition 1.16.1 to graphs of bounded treewidth. The graph operations we use are similar though different from the ones found in the literature (see [12, Section 2.3.1] for example).

A *sourced graph* is a graph with a sequence of some of its vertices called its *sources*. The sequence of sources is called the *interface* of the sourced graph. The number of sources of a sourced graph G is called its *arity* and is denoted by $\mathrm{ar}(G)$.

Given a sourced graph G, the nonsourced graph obtained by the addition to G of every possible edge between its sources, and then by forgetting that G has an interface, is called the *skeleton* of G.

Tree decompositions of sourced graphs are defined as for simple graphs (see Definition 1.14.1), with one further condition: a bag of the decomposition contains all sources. The *treewidth* of a sourced graph is the minimum width of its tree decompositions.

By Lemma 1.14.2, a sourced graph and its skeleton have the same tree decompositions. As a consequence, the treewidth of a sourced graph and of its skeleton are equal.

Since in a tree decomposition of a sourced graph a bag contains all sources, a sourced graph of arity $k + 1$ has treewidth at least k.

Definition 1.16.2. Let G and H be graphs of arity k.

- Denoting the sources of G and H by r_i^G and r_i^H ($i = 1, \ldots, k$), respectively, the *parallel composition* of G and H, is the graph of arity k defined by:

$$G \parallel H = (G \uplus H)/r_1^G r_1^H/\ldots/r_k^G r_k^H.$$

- 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 source 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 making it a source by appending it to G's interface.

- For a permutation p of $[1, k]$, we denote by pG the graph obtained from G by permuting its interface according to p.

The sourced graph $\emptyset_k$ of arity k which is reduced to its interface is called the *empty graph of arity k*.

We have: $\emptyset = \emptyset_0$ and $\emptyset_{k+1} = 1\emptyset_k$.

Proposition 1.16.3. *Let $k \geq 1$ be an integer. The closure of the empty graph and of the edge with two sources under parallel compositions, forgets, lifts (limiting the arity to $k+1$), and permutations is exactly the set of sourced graphs of treewidth at most k.*

Proof. We prove the following two statements of which the proposition is a direct consequence:

1. Bounds on the treewidth are preserved by parallel compositions, forget operations, and permutations of sources. If a graph with up-to k sources has treewidth at most k, then its lift has treewidth at most k.

2. Any non-empty sourced graph of treewidth at most k distinct from the edge with two sources is either the parallel composition of two non-empty sourced graphs of treewidth at most k, or the permutation of a lift of a sourced graph of treewidth at most k, or the forget of a sourced graph of treewidth at most k.

1. A tree decomposition of a sourced graph remains valid for its forget or permutation so bounds on the treewidth are preserved by these two operations.

Given a tree decomposition T for a graph G of arity at most k, one can build a tree decomposition for the lift of G by adding to T a bag b containing precisely the sources of G as well as the new source added by the lift operation, making b adjacent, in T, to a bag of containing all of G's sources. The width of the new tree decomposition remains at most k, thanks to G having at most k sources.

Given two tree decompositions T and T' of two sourced graphs G and H of the same arity, one can build a tree decomposition for their parallel composition by taking the disjoint union T and T', adding a new bag b containing the sources of $G \parallel H$, and making b adjacent to a bag of T (resp. a bag of T') containing all sources of G (resp. H).

2. The fact that in all cases arguments of the operations have treewidth at most k is easily verified.

Let G be a sourced graph of treewidth at most k distinct from the edge with two sources.

If a source s is isolated in G we easily combine a lift adding this source with a permutation to prove that G is the permutation of a lift of $G - s$.

Assume henceforth that every source of G is nonisolated.

If there is an edge e between two sources in G, then G is the parallel composition of $G - e$ with a graph reduced to e and its sources. As sources of G are nonisolated and as G is distinct from the edge with two sources, $G - e$ is non-empty.

Next, consider the case where no edge of G has two sources as endpoints.

Let S be the set of sources of G. If there exists at least two components in $G - S$, then write C for one of them. Having two components in $G - S$ ensures that $G - C$ is non-empty. Then G is the parallel composition of $G[C \cup \{S\}]$ with $G - C$.

Suppose in the rest of the proof there is exactly one component in $G - S$.

First, we prove that G has at most k sources. Let H be the minor of G's skeleton defined by contracting all edges of C. The order of H is $\mathrm{ar}(G) + 1$; its vertices are the sources of G plus one vertex x corresponding to the contraction of C. In H, there is a an edge between x and a source $s \in S$ of G if and only if there is in G an edge between s and a vertex of C. As there is only one component in $G - S$, as sources are nonisolated in G, and as sources are pairwise nonadjacent in G, every source $s \in S$ must be adjacent to a vertex of C in G, and in H, each vertex $s \in S$ is adjacent to x. By definition of skeleton, it holds that vertices of S are pairwise adjacent in H.

Hence, H is $K_{\mathrm{ar}(G)+1}$ and a minor of G's skeleton. By Proposition 1.15.6, as G has treewidth at most k, and hence its skeleton too, we have $\mathrm{ar}(G)+1 < k+2$.

Let T be a tree decomposition of width at most k of G. Let b be a bag of T containing all sources of G. As there is one component in $G-S$, there must be at least one nonsource vertex in G. Consider a shortest path P in T from b to a bag b' containing a nonsource vertex x of G. Let T' be the tree decomposition obtained by adding x to all bags of P. All bags of P but b' contain only sources. In particular, as T has width at most k, all bags of P but b' contain at most k vertices. Hence, bags of T' contain at most $k + 1$ vertices and T' has width at most k.

The tree decomposition T' of G is also a tree decomposition of the sourced graph G' obtained from G by promoting x as a source (appending x to G's interface). By definition, the graph G is the forget of G'. □

Remark 1.16.4. To generate graphs of treewidth at most k, it is also possible to limit the arity to k by considering another operation often called the *series operation*. It is discussed and introduced in § 4.6 as an operation derived from the ones already defined.

CONNECTIVITY DECOMPOSITIONS

A big part of Chapter 4 consists in connectivity decompositions, but adapted to sourced graphs. Here we recall how a graph can be decomposed along some of its vertices whose deletions separate the graph, and the principles behind connectivity-oriented proofs.

1.17 Connectivity. Let k be a non-negative integer. A graph is called *k-connected* if it has at least $k+1$ vertices, and removing at most $k-1$ vertices cannot disconnect it. The *connectivity* of a graph is the maximum integer k such that the graph is k-connected.

For example 0-connected graphs are precisely the non-empty graphs, 1-connected graphs are precisely the non-trivial connected graphs, and every cycle is 2-connected.

In this section we discuss generalisations of the decomposition in connected components of Proposition 1.10.1.

1.18 Cutvertices and blocks. Let $G = (V, E)$ be a graph. A subset $X \subseteq V(G)$ of vertices *separates* a vertex x from a vertex y in G when all x-y paths contain a vertex of X and $x, y \notin X$.

A *cutvertex* (resp. a *bridge*) in G is a vertex $x \in V$ (resp. an edge $e \in E$) whose deletion disconnects two vertices of G. A maximal connected subgraph H of G such that H does not contain a cutvertex is called a *2-component* of G.

The maximal subgraphs of G which are 2-connected are called its *blocks*.

Theorem 1.18.1. *Let G be a graph. We have:*

1. *The intersection of two 2-components C_1 and C_2 of a graph G is either empty, or reduced to a cutvertex c of G separating $C_1 - c$ from $C_2 - c$.*

2. *The 2-components of a graph are exactly its blocks, its bridges, as well as its isolated vertices.*

Proof. We prove each item separatly:

1. Consider two 2-components C_1 and C_2 of G with a non-empty intersection. By maximality of C_1 and C_2, the graph $C_1 \cup C_2$ contains a cutvertex.

   Let u be a vertex in $C_1 \cup C_2$. We prove that if there exists a vertex v common to C_1 and C_2 and distinct from u, then u cannot be a cutvertex of $C_1 \cup C_2$. As a consequence of this fact, the graph $C_1 \cup C_2$ has a unique cutvertex which is the unique vertex in the intersection of C_1 and C_2.

Fix $i \in \{1,2\}$. Let x and y be two vertices of C_i. As C_i does not contain cutvertices, x and y are connected in $C_i - u$ and by extension in $(C_1 \cup C_2) - u$.

Let x and y be vertices of C_1 and C_2, respectively. As v is common to C_1 and C_2, and as C_1 and C_2 do not contain cutvertices, consider P and Q paths in $C_1 - u$ and $C_2 - u$ from x to v and from v to y, respectively. The walk PQ connects x and y in $(C_1 \cup C_2) - u$.

Hence u is not a cutvertex of $C_1 \cup C_2$.

2. We prove that the 2-components of a graph are exactly its blocks, its bridges, as well as its isolated vertices.

   Let H be a 2-component of a graph G.

   If H is 2-connected, then consider K a subgraph of G containing H as a subgraph. By maximality of H, K contains a cutvertex x. Hence K is not 2-connected. So H is a maximal subgraph of G which is 2-connected: H is a block of G.

   Otherwise, H is not 2-connected. There are two possibilities by definition of connectivity. Either H has at least three vertices and removing one vertex from H disconnects H, or H has at most two vertices. The first case contradicts the fact that H has no cutvertices and in the latter case, H consists either of an isolated vertex of G or of an edge xy. In the latter case we prove that xy is a bridge of G.

   For the sake of contradiction, assume xy is not a bridge in G. In particular, in $G - xy$, there exists a path P from x to y. The cycle Pyx is 2-connected and contains the edge xy, a contradiction with the maximality of H. □

This theorem explains how a graph can be decomposed along its cutvertices in different parts which are either somewhat connected, or very simple.

Given a graph G, denote its set of blocks by $\mathcal{B}$, and the union of its cutvertices and of its vertices incident to a bridge by C. The *block graph* of G is the graph with vertex set $C \uplus \mathcal{B}$, and edges cB if $c \in C$ is a vertex of $B \in \mathcal{B}$, and cc' if cc' is a bridge of G.

See Figure 1.14 for an example of block graph.

Theorem 1.18.2. *Block graphs are forests.*

Proof. Let G be a graph and let H denote its block graph. Towards a contradiction, assume H is not a forest. By definition this means there exists a cycle C in H. It alternates between blocks and cutvertices of G (except for

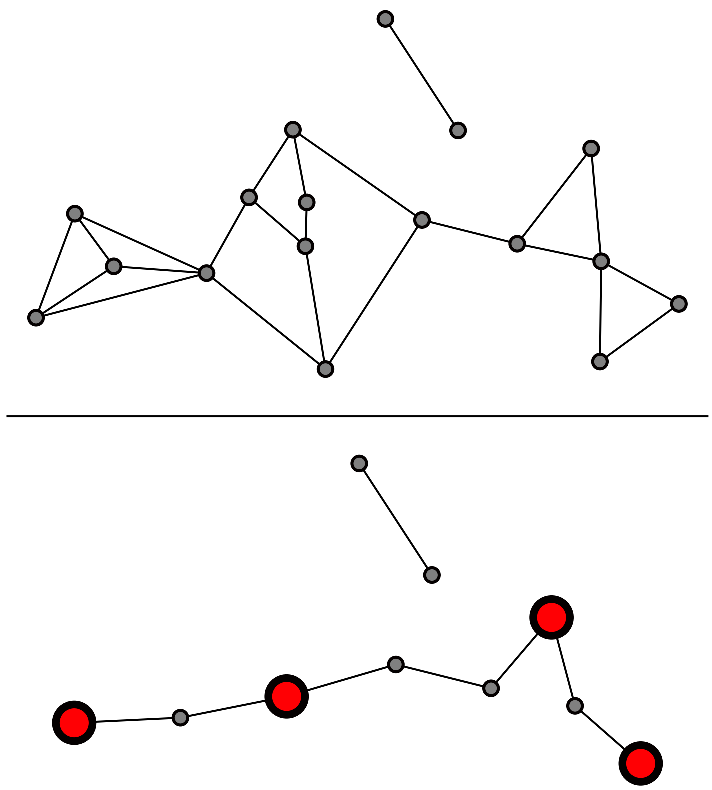


Figure 1.14: The bottom part shows the block graph of the graph on the top part. Block vertices are represented by big red vertices.

some possible bridges). Given a sequence cBc' appearing consecutively on C, c and c' are vertices of B. As B is connected, assume a path P in B from c to c'. Replacing each blocks in C by such paths provides a cycle C' in G.

Assume a sequence BcB' appearing in C. By Theorem 1.18.1, c being common to both B and B', c is a cutvertex in G. The cycle C' shows that vertices of $B - c$ and $B' - c$ are not disconnected in $G - c$, a contradiction with Theorem 1.18.1.

If no sequence BcB' appears in C, then all edges of C are bridges and $C' = C$. Take xy any edge of C. In $G - xy$, as xy is a bridge, x and y are disconnected, a contradiction with the existence of the cycle C. □

Just as for the decomposition of a graph in its connected components, the one of Theorem 1.18.1 in blocks, bridges, and isolated vertices is preserved under isomorphism. More precisely, under isomorphism, cutvertices are mapped to cutvertices, blocks to blocks, bridges to bridges, isolated vertices to isolated vertices, and if two graphs are isomorphic, so are their block graphs. So the decomposition of a graph along its cutvertices is again called *canonical*.

Connectivity decompositions such as above carry on beyond cutvertices, considering separators of size two, three, and four, although, to ensure canonicity, the statements and proofs become very technical, and involve, e.g. variations of 3 and 4-connectivity [45, 8, 27].

1.19 Connectivity-oriented proofs. Many results in graph theory are proved first on sufficiently connected graphs (k-connected for some k). Then, decompositions such as above and their particular structure (disjoint union of connected components, the block graph being a tree, etc) are used to generalise the proof to all graphs.

One of the main results in this thesis is the existence of a finite equational axiomatisation for graphs of treewidth at most three. Its proof follows a connectivity-oriented scheme, but on sourced graphs.

We adapt connectivity decompositions to sourced graphs in Chapter 4, generalisations of connected components and cutvertices being introduced in the form of full prime components and anchors, respectively.

2

Universal algebra

2.1 Introduction. In the first two sections, we present the main tools necessary to formalise *equational reasoning* (§2.8 to §2.11) on *terms* (over a *signature*) interpreted in an *algebra* (§2.2 to §2.5). These notions are standard and discussed in most universal algebra textbooks, such as [48].

We provide examples of graph algebras (§2.6 and §2.7) for forests, graphs of bounded treewidth, and the set of all graphs. See [12] for a general reference on graph algebras (with algebras similar to those we consider).

In the last section (§2.12 to §2.16) we introduce a first example of equational axiomatisation for forests.

ALGEBRAS AND TERMS

2.2 Signatures and algebras. A set X equipped with a function $\mathrm{ar}\colon X \to \mathbb{N}$ from X to the set of non-negative integers is called a *ranked set*. Let x be an element of a ranked set X; the number $\mathrm{ar}(x)$ is called the *arity* of x.

Given a set S whose elements are called *sorts*, a *(functional, multisorted) signature* on S is a ranked set Σ of *symbols* such that to each $f \in \Sigma$ of arity n

is associated a tuple $(s_1, \ldots, s_n)$ of *argument sorts* and a *value sort* $s \in S$; we write $f : s_1 \times \cdots \times s_n \to s$.

Symbols of arity n are called *n-ary*. In particular, symbols of arity zero, one, and two are called *nullary*, *unary*, and *binary*, respectively. Nullary symbols are often called *constants*.

Let Σ be a signature. A *Σ-algebra $\mathcal{A}$* is given by:

- a set E_s for each sort $s \in \Sigma$, called the *domain* of sort s,
- an element $c^{\mathcal{A}} \in E_s$ for each constant c of sort s, and,
- a total function $f^{\mathcal{A}} : E_{s_1} \times \cdots \times E_{s_n} \to E_s$ for each symbol $f : s_1 \times \cdots \times s_n \to s$ of arity $n > 0$.

2.3 First examples. Consider a signature with one sort, one constant, and one binary symbol. The associated algebras are precisely the *pointed magmas*, that is the mathematical structures consisting of a set equipped with a designated element and a binary operation.

A signature for *arithmetic expressions* consists of one sort, two binary symbol $\hat{+}$ and $\hat{*}$, and two constants $\hat{0}$ and $\hat{1}$. Using the set of non-negative integers as a domain, and mapping $\hat{0}$ to 0, $\hat{1}$ to 1, $\hat{+}$ to the addition, and $\hat{*}$ to the multiplication gives the *standard model* of arithmetic expressions.

A priori, there is no reason for $\hat{0}$ to be interpreted as the number 0, or $\hat{+}$ as the addition on integers. We could just as well choose the set of real numbers for the domain, or the set of square matrices over the reals, or even an algebra for which the intuition we have for 0, 1, and addition do not hold (see [25]). Hence different algebras may verify different laws: commutativity of multiplication holds on integers but not on matrices.

2.4 Terms and interpretations. Let Σ and X be two disjoint signatures, such that X contains only nullary symbols called *variables*. The *terms* over $\Sigma \cup X$ are defined recursively as follows:

- a variable $x : s$ in X is a term of sort s,
- a constant $c : s$ in Σ is a term of sort s, and,
- for every symbol $f : s_1 \times \cdots \times s_n \to s$ of arity $n > 0$ and terms t_i of sort s_i ($i = 1, \ldots, n$), $f(t_1, \ldots, t_n)$ is a term of sort s.

The set of all terms over $\Sigma \cup X$ is denoted by $\mathcal{T}_\Sigma(X)$.

The set $\mathcal{T}_\Sigma(X)$ is a Σ-algebra:

- its domain of sort s is the set of terms of sort s,
- $c^{\mathcal{T}_\Sigma(X)} = c$ for every constant c in Σ, and,
- the function $f^{\mathcal{T}_\Sigma(X)}$ associated to a symbol $f\colon s_1 \times \cdots \times s_n \to s$ of arity $n > 0$ acts as follows:

$$f^{\mathcal{T}_\Sigma(X)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n).$$

We denote $\mathcal{T}_\Sigma(\emptyset)$ by $\mathcal{T}_\Sigma$, and call its elements the *closed terms* over the signature Σ.

Let $\mathcal{A}$ be a Σ-algebra. An *interpretation* of X in $\mathcal{A}$ is a sort-preser- ving function σ from X to the domains of $\mathcal{A}$. It extends recursively to a map $\hat{\sigma}$ from $\mathcal{T}_\Sigma(X)$ to the domains of $\mathcal{A}$ by:

$$\hat{\sigma}(f(t_1, \ldots, t_n)) = f^{\mathcal{A}}(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_n)).$$

2.5 Backus-Naur form. Often, signatures and their terms are expressed as context-free grammars in *Backus-Naur form*. We keep this notion informal. For example, for arithmetic expressions we might have introduced the signature as:

$$t := t \,\hat{+}\, t \mid \hat{0} \mid \hat{1}$$

where the different occurrences of t do not denote a common term; they are rather an abstract representation of terms in general.

In the case of a many-sorted signature, each sort is detailed in a single line:

$$\begin{aligned} t_s &:= f(t_{s_1}, \ldots, t_{s_n}) \mid \ldots \\ t_{s'} &:= g(t_{s'_1}, \ldots, t_{s'_m}) \mid \ldots \\ &\ldots \end{aligned}$$

for symbols $f\colon s_1 \times \cdots \times s_n \to s$ and $g\colon s'_1 \times \cdots \times s'_m \to s'$ of arity n and m, respectively.

2.6 A signature for trees and forests. In this paragraph we detail a first (multisorted) example on (rooted) graphs.

Consider the signature Σ_{forest} with two sorts $\mathbf{r}$ and $\mathbf{nr}$, and five symbols:

$$\begin{aligned} \Sigma_{\text{forest}} = \{ \quad \emptyset &: \mathbf{nr}, \\ \mathrm{rl}_{\mathbf{nr}} &: \mathbf{nr} \to \mathbf{r}, \\ \mathrm{rl}_{\mathbf{r}} &: \mathbf{r} \to \mathbf{r}, \\ \parallel &: \mathbf{r} \times \mathbf{r} \to \mathbf{r}, \\ \mathrm{f} &: \mathbf{r} \to \mathbf{nr} \quad \}. \end{aligned}$$

In Backus-Naur form:

$$t_{\mathbf{nr}} := \mathrm{f}t_{\mathbf{r}} \mid \emptyset$$
$$t_{\mathbf{r}} := \mathrm{rl}_{\mathbf{r}}t_{\mathbf{r}} \mid \mathrm{rl}_{\mathbf{nr}}t_{\mathbf{nr}} \mid t_{\mathbf{r}} \parallel t_{\mathbf{r}}.$$

The sorts **r** and **nr** stand for "rooted" and "non-rooted", respectively.

The graph algebra we consider for this signature is the following: the domains of sorts **r** and **nr** are the sets of rooted and non-rooted simple graphs, respectively; f is the forget of a rooted graph, $\emptyset$ is the empty graph, $\mathrm{rl}_{\mathbf{r}}$ and $\mathrm{rl}_{\mathbf{nr}}$ are the root lifts, respectively on rooted and non-rooted graphs, and $\parallel$ is the parallel composition of two rooted graphs.

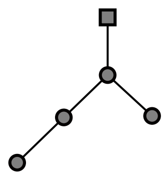
The unique interpretation of the empty set in $\mathfrak{G}_{\text{graph}}$ extends to a map $\mathcal{G}$ from $\mathcal{T}_{\Sigma_{\text{forest}}}$ the set of closed terms to the set of (possibly rooted) graphs.

The symbol $\parallel$ denotes both a graph operation on rooted graphs and a symbol. Although they are formally distinct, henceforth we do not make a distinction between symbols and associated functions in algebras. Hence f also denotes the forget operation of rooted graphs.

A direct corollary of Proposition 1.16.1 is the following:

Corollary 2.6.1. *The set $\mathcal{G}(\mathcal{T}_{\Sigma_{\text{forest}}})$ is precisely that of (possibly rooted) forests.*

Example 2.6.2. The term $\mathrm{rl}_{\mathbf{r}}(\mathrm{rl}_{\mathbf{r}}\mathrm{rl}_{\mathbf{nr}}\emptyset \parallel \mathrm{rl}_{\mathbf{r}}\mathrm{rl}_{\mathbf{r}}\mathrm{rl}_{\mathbf{nr}}\emptyset)$ is interpreted as the tree shown below:



2.7 A signature for simple graphs of bounded treewidth. We introduce a signature Σ_{graph} whose symbols correspond to the graph operations from Definition 1.16.2.

The sorts of Σ_{graph} are the non-negative integers. With k ranging over all non-negative integers, and p over the permutations of $[1,\ldots,k]$, we define:

$$\begin{aligned}\Sigma_{\text{graph}} = \{ \quad \parallel &: k \times k \to k \\ \mathrm{l} &: k \to k+1 \\ \mathrm{f} &: k+1 \to k \\ p &: k \to k \\ \emptyset_0 &: 0 \\ e &: 2 \quad \}.\end{aligned}$$

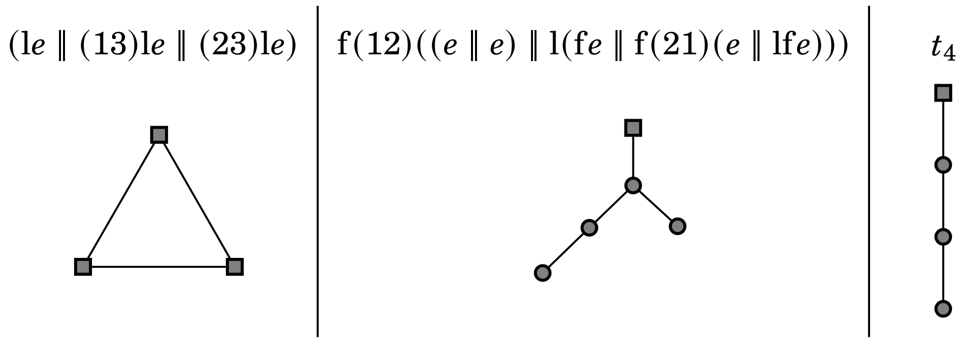


Figure 2.1: Three graphs and one of their parsing. The notation (ij) represents the permutation swapping i and j. The terms t_n are defined by induction by $t_0 = \mathrm{l}\emptyset_0$ and $t_{n+1} = \mathrm{f}(12)(\mathrm{l}t_n \parallel e)$; t_n is a parsing of the the path of length n in which one endpoint is a source.

In Backus-Naur form:

$$\begin{array}{lllll} t_0 = & \mathrm{f}t_1 & \mid pt_0 & \mid t_0 \parallel t_0 & \mid \emptyset_0 \\ t_2 = \mathrm{l}t_1 & \mid \mathrm{f}t_3 & \mid pt_2 & \mid t_2 \parallel t_2 & \mid e \\ t_k = \mathrm{l}t_{k-1} & \mid \mathrm{f}t_{k+1} & \mid pt_k & \mid t_k \parallel t_k. & \end{array}$$

For $k \geq 1$ we define the terms $\emptyset_k$ recursively by $\emptyset_k = \mathrm{l}\emptyset_{k-1}$.

This syntax matches the operations of Definition 1.16.2. Accordingly, there is a function $\mathcal{G}$ from $\mathcal{T}_{\Sigma_{\text{graph}}}$ to the set of sourced graphs; $\mathcal{G}$ preserves arities and the constant e is interpreted as the edge with two sources.

We say that a term t *denotes* a graph G, or that t is a *parsing* of G, when $\mathcal{G}(t) \simeq G$.

Figure 2.1 shows examples of parsings of graphs.

In the syntax induced by Σ_{graph}, vertex additions can only be done using the lift operation: during the construction of a graph via a parsing of it, any vertex must first be added as a source. Hence, the number of inner vertices of a graph equals 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 connected to mean that its graph is so, or calling *arity* the sort of a term, that is, the arity of its graph.

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

A direct corollary of Proposition 1.16.3 is:

Corollary 2.7.1. *A graph has treewidth at most k if and only if it has a parsing of width at most k.*

The signature Σ_{graph} restricted to the sorts $0, \ldots, k+1$ is denoted by Σ^k_{graph}.

Corollary 2.7.1 shows that in the algebra of sourced graphs, the set of closed terms over Σ^k_{graph} is interpreted as the set of graphs of treewidth at most k.

EQUATIONAL REASONING

In an algebra, terms are sometimes interpreted as the same element. For example, with t and u two arithmetic expressions, in the standard model, $t \hat{+} u$ and $u \hat{+} t$ are interpreted as the same integer.

We introduce equations on terms and show how to use them to deduce equalities.

2.8 Equations and algebras. Let Σ be a signature. Henceforth, we assume a signature $\mathcal{V}$ of nullary symbols containing a countable number of variables $x, y, z, \ldots$ of each sort s.

An *equation* is a pair (t, u) of terms in $\mathcal{T}_\Sigma(\mathcal{V})$. We write $t = u$ for the equation (t, u). In this thesis, equations are often referred to as *axioms*.

For example, the axiom $x \hat{+} y = y \hat{+} x$ on arithmetic expressions expresses *commutativity* of addition.

Let $t = u$ be an axiom and $\mathcal{A}$ a Σ-algebra. We say that $t = u$ is *verified* in $\mathcal{A}$ when, for every interpretation $\sigma \colon \mathcal{V} \to \mathcal{A}$ of the variables, $\hat{\sigma}(t) = \hat{\sigma}(u)$ holds.

Intuitively, the variables of an equation are understood as universally quantified.

2.9 Monoids. Consider Σ_{magma} the relational signature of pointed magmas: it contains one sort, one constant e, and one binary operation $\cdot$. By definition, the algebras in which the following three axioms are verified are precisely *monoids*:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$
$$x \cdot e = x$$
$$e \cdot x = x$$

that is, e is a *neutral element*, and $\cdot$ is *associative*.

From these three axioms follow many other equations on monoids, in the usual manner. For example,

$$\begin{aligned}(x \cdot e) \cdot (e \cdot (z \cdot y)) &= x \cdot (e \cdot (z \cdot y)) \\ &= x \cdot (z \cdot y) \\ &= (x \cdot z) \cdot y.\end{aligned}$$

In the next paragraph we formalise the associated notion of *equational calculus*.

In a monoid some equations may be verified and not deducible from the monoid axioms. For example some monoids are commutative. A natural problem is the following: for which Σ_{magma}-algebras the verified equations are precisely captured by the monoid axioms? This question has a simple answer in the form of *free monoids*, meaning, roughly, the monoids A^* defined as the words on a set A of letters, with $\cdot$ being word concatenation and e the empty word.

This thesis focuses on the inverse question: for each of the introduced graph signatures, is there a (finite) set of axioms such that the equations verified by the associated graph algebra are precisely captured by the axioms?

2.10 Equational calculus. Let S be a set of sort, Σ an associated signature, and $\mathcal{V}$ a signature of variables. For each sort $s \in S$, we suppose that $\mathcal{V}$ contains a particular variable referred to as the *hole* of sort s, and denoted by $[]_s$.

A *(term) substitution* is an interpretation of variables to terms, that is, a function $\sigma\colon \mathcal{V} \to \mathcal{T}_\Sigma(\mathcal{V})$. such that:

- $\sigma([]_s) = []_s$ for all holes, and,
- if x is not a hole, then no holes appear in $\sigma(x)$.

A *context* is a term C in which there is precisely one hole. Let σ be the variable interpretation mapping the hole of C to a term u of the same sort, and defined as the identity otherwise. The *substitution of the hole* by a term u in C is defined by $\hat{\sigma}(C)$ and is denoted by $C[u]$.

Given a set of axioms Ax, we define the associated *equational calculus* as the least reflexive and transitive relation $\equiv_{\text{Ax}}$ on terms such that for every context C, substitution σ, and equation $t = u$ or $u = t$ in Ax,

$$C[\hat{\sigma}(t)] \equiv_{\text{Ax}} C[\hat{\sigma}(u)].$$

Hence two terms t and u are equated by $\equiv_{\mathrm{Ax}}$ whenever $t \equiv_{\mathrm{Ax}} u$ can be proved by repeatedly applying axioms in Ax, up to contexts and substitutions:

$$\begin{array}{lll} t & = & C_1[\hat{\sigma}_1(t_1)] \\ & \equiv_{\mathrm{Ax}} & C_1[\hat{\sigma}_1(u_1)] \qquad ((t_1, u_1) \in \mathrm{Ax}) \\ & = & C_2[\hat{\sigma}_2(t_2)] \\ & \equiv_{\mathrm{Ax}} & C_2[\hat{\sigma}_2(u_2)] \qquad ((t_2, u_2) \in \mathrm{Ax}) \\ & \dots & \\ & = & C_n[\hat{\sigma}_n(t_n)] \\ & \equiv_{\mathrm{Ax}} & C_n[\hat{\sigma}_n(u_n)] \qquad ((t_n, u_n) \in \mathrm{Ax}) \\ & = & u \end{array}$$

An example is provided for monoids in the previous paragraph.

2.11 Equational theories. An *equational theory* R is an equivalence relation on terms relating only terms of the same sort, and that is stable under substitution and context. Formally, it is an equivalence relation preserving sorts and such that whenever $t\ R\ u$:

- if C is a context then $C[t]\ R\ C[u]$ and
- if σ is a substitution, then $\hat{\sigma}(t)\ R\ \hat{\sigma}(u)$.

The equational theory Eq(Ax) associated to a set of axiom Ax is the least equational theory containing Ax.

Theorem 2.11.1. *The equational theory and the equational calculus associated to a set of axioms coincide:*

$$\mathrm{Eq}(\mathrm{Ax}) = \equiv_{\mathrm{Ax}}.$$

Proof. We prove both inclusion separately.

- $\mathrm{Eq}(\mathrm{Ax}) \supseteq \equiv_{\mathrm{Ax}}$; let t and u be two terms such that $t \equiv_{\mathrm{Ax}} u$. We show t Eq(Ax) u by induction on $\equiv_{\mathrm{Ax}}$.
  - **Reflexivity;** if $t = u$ then t Eq(Ax) u holds as Eq(Ax) is an equivalence relation.
  - **Transitivity;** assume there exists a term v such that $t \equiv_{\mathrm{Ax}} v$ and $v \equiv_{\mathrm{Ax}} u$. By induction hypothesis, t Eq(Ax) v and v Eq(Ax) u hold. By transitivity of Eq(Ax), t Eq(Ax) u holds.

- **Axioms;** assume that for some axiom $t' = u'$ or $u' = t'$ in Ax, we have $t = C[\hat{\sigma}(t')]$ and $C[\hat{\sigma}(u')] = u$.
  By definition Eq(Ax) contains axioms. Moreover, Eq(Ax) is symmetric. Thus t' Eq(Ax) u'.
  Since Eq(Ax) is an equational theory, t Eq(Ax) u.

- Eq(Ax) $\subseteq \equiv_{\text{Ax}}$; we prove that $\equiv_{\text{Ax}}$ is an equational theory. As Ax $\subseteq \equiv_{\text{Ax}}$ and since Eq(Ax) is the least equational theory containing Ax, the inclusion follows.

  The relation $\equiv_{\text{Ax}}$ is symmetric as the rule $C[\hat{\sigma}(t)] \equiv_{\text{Ax}} C[\hat{\sigma}(u)]$ can be applied for either $t = u$ or $u = t$ an axiom.

  Hence, $\equiv_{\text{Ax}}$ is an equivalence relation.

  Let C be a context and t and u two terms such that $t \equiv_{\text{Ax}} u$. We prove that $C[t] \equiv_{\text{Ax}} C[u]$ by induction on $\equiv_{\text{Ax}}$:

  - **Reflexivity;** if $t = u$ then $C[t] = C[u]$ and by reflexivity $C[t] \equiv_{\text{Ax}} C[u]$.
  - **Transitivity;** let v be a term such that $t \equiv_{\text{Ax}} v$ and $v \equiv_{\text{Ax}} u$. By induction hypothesis, $C[t] \equiv_{\text{Ax}} C[v]$ and $C[v] \equiv_{\text{Ax}} C[u]$. Since $\equiv_{\text{Ax}}$ is transitive, we get $C[t] \equiv_{\text{Ax}} C[u]$.
  - **Axioms;** let C' be a context, σ a substitution, and $t' = u'$ and axiom in Ax such that $t = C'[\hat{\sigma}(t')]$ and $C'[\hat{\sigma}(u')] = u$. The term $C[C']$ is a context. Hence
  $$C[t] = C[C'][\hat{\sigma}(t')] \equiv_{\text{Ax}} C[C'][\hat{\sigma}(u')] = u.$$

  Let σ be a substitution and t and u two terms such that $t \equiv_{\text{Ax}} u$. We prove $\hat{\sigma}(t) \equiv_{\text{Ax}} \hat{\sigma}(u)$ by induction on $\equiv_{\text{Ax}}$:

  - **Reflexivity;** if $t = u$ then $\hat{\sigma}(t) = \hat{\sigma}(u)$ and by reflexivity $\hat{\sigma}(t) \equiv_{\text{Ax}} \hat{\sigma}(u)$.
  - **Transitivity;** let v be a term such that $t \equiv_{\text{Ax}} v$ and $v \equiv_{\text{Ax}} u$. By induction hypothesis, $\hat{\sigma}(t) \equiv_{\text{Ax}} \hat{\sigma}(v)$ and $\hat{\sigma}(v) \equiv_{\text{Ax}} \hat{\sigma}(u)$. Since $\equiv_{\text{Ax}}$ is transitive, we get $\hat{\sigma}(t) \equiv_{\text{Ax}} \hat{\sigma}(u)$.
  - **Axioms;** let C be a context, σ' a substitution, and $t' = u'$ and axiom in Ax such that $t = C[\hat{\sigma}'(t')]$ and $C[\hat{\sigma}'(u')] = u$. Since σ is the identity on holes, and since holes do not appear in the image of σ, we have that $\hat{\sigma}(C)$ is a context, and
  $$\hat{\sigma}(t) = (\hat{\sigma}(C))[(\hat{\sigma} \circ \sigma')(t)]$$
  and similarly for u, proving $\hat{\sigma}(t) \equiv_{\text{Ax}} \hat{\sigma}(u)$. □

Since $\equiv_{\mathrm{Ax}}$ is an equational theory, we can deduce new equalities using already deduced ones. Indeed, if $t \equiv_{\mathrm{Ax}} u$, then for any context C and substitution σ, we have:

$$C[\hat{\sigma}(t)] \equiv_{\mathrm{Ax}} C[\hat{\sigma}(u)].$$

AN AXIOMATISATION OF FORESTS

As a first example of finite axiomatisation on graphs, we show that there exists a finite set Ax of axioms on the signature Σ_{forest} such that for every pair of closed terms $t, u \in \mathcal{T}_{\Sigma_{\mathrm{forest}}}$,

$$\mathcal{G}(t) \simeq \mathcal{G}(u) \quad \Leftrightarrow \quad t \equiv_{\mathrm{Ax}} u.$$

We show how graph equational reasoning should be approached: term manipulations via axioms can be understood as successive zooms (in and out) on the corresponding graphs.

2.12 Isomorphism of rooted graphs. Two rooted graphs G and H with respective roots r_G and r_H are called *isomorphic* when there is a graph *isomorphism* $\phi\colon V(G) \to V(H)$ preserving roots: $\phi(r_G) = r_H$. As in §1.6 we write $G \simeq H$ when two rooted graphs are isomorphic.

2.13 Forest axioms. Let $\mathrm{Ax}_{\mathrm{forest}}$ be the set containing the following seven axioms:

A1. $x \parallel (y \parallel z) = (x \parallel y) \parallel z$,

A2. $x \parallel y = y \parallel x$,

A3. $x \parallel \mathrm{rl}_{\mathbf{nr}}\emptyset = x$,

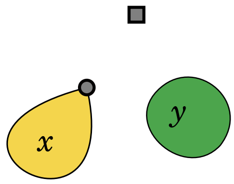
A4. $\mathrm{f}(\mathrm{rl}_{\mathbf{r}}x \parallel y) = \mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{r}}y)$,

A5. $\mathrm{f}(\mathrm{rl}_{\mathbf{nr}}\mathrm{f}x \parallel y) = \mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{nr}}\mathrm{f}y)$,

A6. $\mathrm{rl}_{\mathbf{r}}(x \parallel \mathrm{rl}_{\mathbf{nr}}y) = \mathrm{rl}_{\mathbf{r}}x \parallel \mathrm{rl}_{\mathbf{nr}}y$,

A7. $\mathrm{rl}_{\mathbf{nr}}(\mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{nr}}y)) = \mathrm{rl}_{\mathbf{nr}}\mathrm{f}x \parallel \mathrm{rl}_{\mathbf{nr}}y$.

To understand intuitively what these axioms represent, we construct the graphs described by the corresponding terms. For example, both terms of $\mathrm{rl}_{\mathbf{nr}}(\mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{nr}}y)) = \mathrm{rl}_{\mathbf{nr}}\mathrm{f}x \parallel \mathrm{rl}_{\mathbf{nr}}y$ describe a graph of the following shape:



and the axiom states that y can be put in parallel either before or after performing a forget and a root lift on x.

We write $\equiv$ for the restriction to $\mathcal{T}_{\Sigma_{\text{forest}}}$ of the equational calculus on terms in $\mathcal{T}_{\Sigma_{\text{forest}}}(\mathcal{V})$ induced by the forest axioms in $\text{Ax}_{\text{forest}}$.

2.14 Soundness of the forest axioms. We prove the *soundness* of the forest axioms:

$$\forall t, u \in \mathcal{T}_{\Sigma_{\text{forest}}},\ t \equiv u \Rightarrow \mathcal{G}(t) \simeq \mathcal{G}(u).$$

To simplify the proof we start by showing how contexts and substitutions relate to forest isomorphism.

Lemma 2.14.1. *Let C be a context over Σ_{forest} whose sole variable is its hole, and $t, u \in \mathcal{T}_{\Sigma_{\text{forest}}}$ two closed terms. If $\mathcal{G}(t) \simeq \mathcal{G}(u)$, then $\mathcal{G}(C[t]) \simeq \mathcal{G}(C[u])$.*

Proof. We proceed by induction on the definition of C.

- **$C = []$ is reduced to a hole;** we have $C[t] = t$ and $C[u] = u$ and $\mathcal{G}(t) \simeq \mathcal{G}(u)$ by hypothesis.
- **$C = \text{rl}_x C'$ for a context C' and $x \in \{\mathbf{r}, \mathbf{nr}\}$;** the graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are isomorphic to $\text{rl}_x\mathcal{G}(C'[t])$ and $\text{rl}_x\mathcal{G}(C'[u])$, respectively.

  By induction hypothesis, $\mathcal{G}(C'[t]) \simeq \mathcal{G}(C'[u])$. Let G be the graph $\mathcal{G}(C'[t])$. The graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are both isomorphic to $\text{rl}_x G$.
- **$C = v \parallel C'$ for a closed term v and a context C';** the graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are isomorphic to $\mathcal{G}(v) \parallel \mathcal{G}(C'[t])$ and to $\mathcal{G}(v) \parallel \mathcal{G}(C'[u])$, respectively.

  By induction hypothesis, $\mathcal{G}(C'[t]) \simeq \mathcal{G}(C'[u])$. Let G be the graph $\mathcal{G}(C'[t])$. The graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are both isomorphic to $\mathcal{G}(v) \parallel G$.
- **$C = \text{f}C'$ for a context C';** the graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are isomorphic to $\text{f}\mathcal{G}(C'[t])$ and $\text{f}\mathcal{G}(C'[u])$, respectively.

  By induction hypothesis, $\mathcal{G}(C'[t]) \simeq \mathcal{G}(C'[u])$. Let G be the graph $\mathcal{G}(C'[t])$. The graphs $\mathcal{G}(C[t])$ and $\mathcal{G}(C[u])$ are both isomorphic to $\text{f}G$. □

Lemma 2.14.2. *Let σ be a substitution and $t = u$ and axiom in* $\mathrm{Ax}_{\mathrm{forest}}$. *If the image of every variable appearing in either t or u under σ is a closed term, then $\mathcal{G}(\hat{\sigma}(t)) \simeq \mathcal{G}(\hat{\sigma}(u))$.*

Proof sketch. This proof requires verifying that the axioms reflect algebraic properties of the graphs and graph operations corresponding to the symbols of Σ_{forest}.

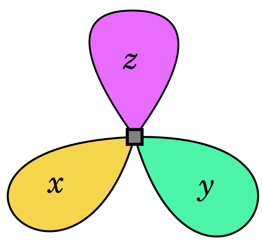
More precisely, we have, for any closed terms t' and u',

- $\mathcal{G}(\emptyset) = \emptyset$,
- $\mathcal{G}(\mathrm{rl}_{\mathbf{nr}} t') \simeq \mathrm{rl}_{\mathbf{nr}} \mathcal{G}(t')$,
- $\mathcal{G}(\mathrm{rl}_{\mathbf{r}} t') \simeq \mathrm{rl}_{\mathbf{r}} \mathcal{G}(t')$,
- $\mathcal{G}(t' \parallel u') \simeq \mathcal{G}(t') \parallel \mathcal{G}(u')$,
- $\mathcal{G}(\mathrm{f} t') \simeq \mathrm{f} \mathcal{G}(t')$,

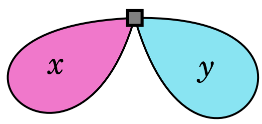
Hence, given a term t' and a substitution σ, $\mathcal{G}(\hat{\sigma}(t'))$ can be expressed by interpreting (in t') symbols in Σ_{forest} as the constant graphs and graph operations of the graph algebra of (possibly rooted) forests, substituting in t' every variable x with $\mathcal{G}(\hat{\sigma}(x))$

We depict, for each axiom in $\mathrm{Ax}_{\mathrm{forest}}$, the graph shape, obtained by the procedure just described, from which $\mathcal{G}(\hat{\sigma}(t)) \simeq \mathcal{G}(\hat{\sigma}(u))$ can be retrieved.

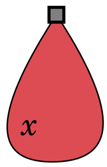
- $x \parallel (y \parallel z) = (x \parallel y) \parallel z$;



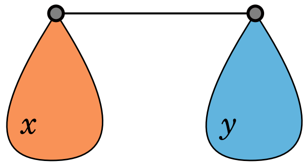
- $x \parallel y = y \parallel x$;



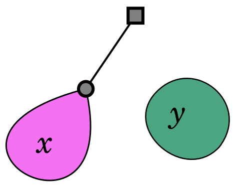
- $x \parallel \mathrm{rl}_{\mathbf{nr}} \emptyset = x$;



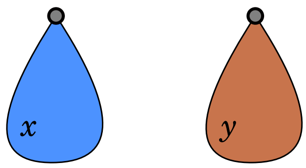
- $\mathrm{f}(\mathrm{rl}_{\mathbf{r}}x \parallel y) = \mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{r}}y)$;



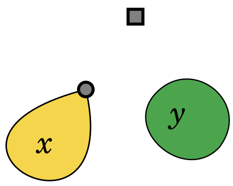
- $\mathrm{f}(\mathrm{rl}_{\mathbf{nr}}\mathrm{f}x \parallel y) = \mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{nr}}\mathrm{f}y)$;



- $\mathrm{rl}_{\mathbf{r}}(x \parallel \mathrm{rl}_{\mathbf{nr}}y) = \mathrm{rl}_{\mathbf{r}}x \parallel \mathrm{rl}_{\mathbf{nr}}y$;



- $\mathrm{rl}_{\mathbf{nr}}(\mathrm{f}(x \parallel \mathrm{rl}_{\mathbf{nr}}y)) = \mathrm{rl}_{\mathbf{nr}}\mathrm{f}x \parallel \mathrm{rl}_{\mathbf{nr}}y$;



□

Proposition 2.14.3 (Soundness of $\mathrm{Ax}_{\mathrm{forest}}$)**.** *For every pair of terms $t, u \in \mathcal{T}_{\Sigma_{\mathrm{forest}}}$,*

$$t \equiv u \Rightarrow \mathcal{G}(t) \simeq \mathcal{G}(u).$$

Proof. Let $t, u \in \mathcal{T}_{\Sigma_{\text{forest}}}$ be two terms such that $t \equiv u$.

We prove $\mathcal{G}(t) \simeq \mathcal{G}(u)$ by induction on $\simeq$:

- **Reflexivity;** if $t = u$, then the graphs $\mathcal{G}(t)$ and $\mathcal{G}(u)$ are equal and hence isomorphic.
- **Transitivity;** let v be a term such that $t \equiv v$ and $v \equiv u$. By induction hypothesis, $\mathcal{G}(t) \simeq \mathcal{G}(v)$ and $\mathcal{G}(v) \simeq \mathcal{G}(u)$. By transitivity of graph isomorphism, $\mathcal{G}(t) \simeq \mathcal{G}(u)$.
- **Axioms;** let C, σ, and $t' = u'$ be a context, a substitution, and an axiom in $\text{Ax}_{\text{forest}}$ such that:

  $$t = C[\hat{\sigma}(t')] \qquad\qquad C[\hat{\sigma}(u')] = u.$$

  Since, t and u are closed terms, the substitution of any variable appearing in either t' or u' is mapped to a closed term by σ, and, except for its hole, every variable of C is a variable of both u and t.

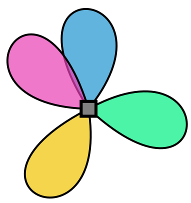
  By Lemmas 2.14.2 and 2.14.1, we have $\mathcal{G}(t) \simeq \mathcal{G}(u)$. □

2.15 Completeness of the forest axioms. We prove the *completeness* of the forest axioms:

$$\forall t, u \in \mathcal{T}_{\Sigma_{\text{forest}}},\ t \equiv u \Leftarrow \mathcal{G}(t) \simeq \mathcal{G}(u).$$

We first prove three lemmas showing how terms in $\mathcal{T}_{\Sigma_{\text{forest}}}$ can be decomposed along the structure of forests.

A tree T with root r can be decomposed in its *subtrees*, meaning the components of $T - r$. In other words, a rooted tree has the following shape:



where the coloured areas represent the subtrees.

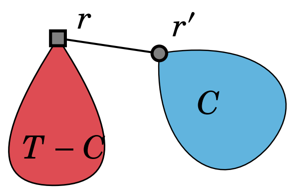
The following lemma shows that, if we choose a given subtree C of a tree T, then C can be isolated at the level of terms using axioms.

Lemma 2.15.1. *Let T be a rooted tree with root r. Let r' be the neighbour of r in a component C of $T - r$.*

For every parsing t of T there exists parsings t^C of C (with r' the root) and t^{T-C} of $T - C$ such that:

$$t \equiv t^{T-C} \parallel \text{rl}_{\mathbf{r}} t^C.$$

Proof. The statement implies that T has shape:



We reason by induction on the structure of the term t.

- $t = \mathrm{rl}_{\mathbf{r}} v$; this implies that r has only one neighbour in T so that v is a parsing of C in which r' is the root. By setting $t^C = v$, $t^{T-C} = \mathrm{rl}_{\mathbf{nr}}\emptyset$ we get:

$$\begin{aligned} t &= \mathrm{rl}_{\mathbf{r}} v \\ &\equiv \mathrm{rl}_{\mathbf{r}} v \parallel \mathrm{rl}_{\mathbf{nr}}\emptyset && \text{(A3)} \\ &\equiv \mathrm{rl}_{\mathbf{nr}}\emptyset \parallel \mathrm{rl}_{\mathbf{r}} v && \text{(A2)} \\ &= t^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} t^C \end{aligned}$$

- $t = \mathrm{rl}_{\mathbf{nr}} v$; this is not possible as then T's root would be isolated, a contradiction with the existence of C.

- $t = u \parallel v$; the component C of $T - r$ must be a component of either $\mathcal{G}(u) - r$ or of $\mathcal{G}(v) - r$. Up-to using A2, we can assume without loss of generality that C is a component of $\mathcal{G}(v) - r$.

  We apply the induction hypothesis to v. We get parsings v^{T-C} and t^C of $\mathcal{G}(v) - C$ and of C such that $v \equiv v^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} t^C$. We conclude as follows, with $t^{T-C} = u \parallel v^{T-C}$:

$$\begin{aligned} t &= u \parallel v \\ &\equiv u \parallel (v^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} t^C) \\ &\equiv (u \parallel v^{T-C}) \parallel \mathrm{rl}_{\mathbf{r}} t^C && \text{(A1)} \\ &= t^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} t^C \end{aligned}$$

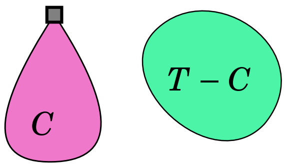
□

Lemma 2.15.2. *Let T be a forest, and C the component of T containing its root.*

For every parsing t of T, there exists parsings t^C of C and t^{T-C} of $T - C$ such that:

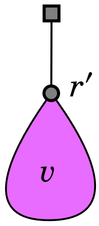
$$t \equiv t^C \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C}.$$

Proof. This statement implies that T has the following shape:



We reason by induction on the structure of the term t:

- $t = \mathrm{rl}_{\mathbf{r}} v$; T has the following shape:



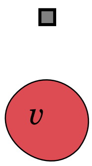
with r' is the root of $\mathcal{G}(v)$, which is also the unique neighbour of T's root. Write C' for $C - r$. Observe that $\mathcal{G}(v) - C' = T - C$. By induction hypothesis there exists parsings $v^{C'}$ and t^{T-C} of C' (with root r') and $T - C$, respectively, such that

$$v \equiv v^{C'} \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C}.$$

We take $t^C = v^{C'}$ and conclude as follows:

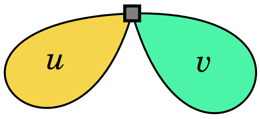
$$\begin{aligned} t &= \mathrm{rl}_{\mathbf{r}} v \\ &\equiv \mathrm{rl}_{\mathbf{r}}(v^{C'} \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C}) \\ &\equiv \mathrm{rl}_{\mathbf{r}} v^{C'}_{\mathbf{r}} \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C} && \text{(A6)} \\ &= t^C \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C} \end{aligned}$$

- $t = \mathrm{rl}_{\mathbf{nr}} v$; this equality shows that T has shape:



Hence C is reduced to the root. We set $t^C = \mathrm{rl}\emptyset$ and $t^{T-C} = v$ and conclude with A3.

- $t = u \parallel v$; T is the disjoint union of $\mathcal{G}(u)$ and $\mathcal{G}(v)$ in which the roots of those two graphs are identified. Hence T has the following shape:



There exists components C_u and C_v of $\mathcal{G}(u)$ and $\mathcal{G}(v)$ such that $C = C_u \parallel C_v$.

We apply the induction hypothesis on both u and v. We get parsings u^C, u^{T-C}, v^C, and v^{T-C}, of C_u, $\mathcal{G}(u) - C_u$, C_v, and $\mathcal{G}(v) - C_v$ such that $u \equiv u^C \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}$ and $v \equiv v^C \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}$.

We take $t^C = u^C \parallel v^C$.

If both $\mathcal{G}(u) - C_u$ and $\mathcal{G}(v) - C_v$ are empty, then $u^{T-C} = \emptyset = v^{T-C}$ (indeed $\emptyset$ is the only possible parsing of the empty graph since a forget would imply at least one vertex). We conclude with $t^{T-C} = \mathrm{rl}_{\mathbf{nr}}\emptyset$:

$$\begin{aligned}
t &= u \parallel v \\
&\equiv (u^C \parallel \mathrm{rl} u^{T-C}) \parallel (v^C \parallel \mathrm{rl} v^{T-C}) \\
&= (u^C \parallel \mathrm{rl}\emptyset) \parallel (v^C \parallel \mathrm{rl}\emptyset) \\
&\equiv u^C \parallel v^C && \text{(A3)} \\
&\equiv (u^C \parallel v^C) \parallel \mathrm{rl}_{\mathbf{nr}}\emptyset && \text{(A3)} \\
&= t^C \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C}
\end{aligned}$$

Otherwise one of $\mathcal{G}(u) - C_u$ or $\mathcal{G}(v) - C_v$ is non-empty. Thanks to A2, assume, without loss of generality that $\mathcal{G}(u) - C_u$ is non-empty. There exists a term w^{T-C} such that $u^{T-C} = \mathrm{f} w^{T-C}$. We conclude with $t^{T-C} = \mathrm{f}(w^{T-C} \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C})$:

$$\begin{aligned}
t &= u \parallel v \\
&\equiv (u^C \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}) \parallel (v^C \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}) \\
&\equiv (u^C \parallel v^C) \parallel (\mathrm{rl}_{\mathbf{nr}} u^{T-C} \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}) && \text{(A1,2)} \\
&= (u^C \parallel v^C) \parallel (\mathrm{rl}_{\mathbf{nr}} \mathrm{f} w^{T-C} \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}) \\
&\equiv (u^C \parallel v^C) \parallel \mathrm{rl}_{\mathbf{nr}}(\mathrm{f} w^{T-C} \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}) && \text{(A7)} \\
&= t^C \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C} && \square
\end{aligned}$$

Lemma 2.15.3. *Let x be a vertex of a non-rooted forest T.*

For every parsing t of T, there exists a parsing t^x of T with root x such that:

$$t \equiv \mathrm{f}t^x$$

Proof. As T is non-empty, there exists a term u such that $t = \mathrm{f}u$. Let r be the root of $\mathcal{G}(u)$.

First, we show that there exists a term u' whose root is in the same component as x and such that $t = \mathrm{f}u'$.

We proceed by induction on the order $|T|$. If r and x are in the same component in T then we set $u' = u$. Otherwise, let C be the component of T containing r. By, Lemma 2.15.2 there exists parsings v^C and v^{T-C} of C and $T - C$ such that $u \equiv v^C \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}$.

By induction hypothesis, there exists a parsing v' of $T - C$ with a root in the same component as x and such that $v^{T-C} \equiv \mathrm{f}v'$.

We conclude with $u' = \mathrm{rl}_{\mathbf{nr}}\mathrm{f}v^C \parallel v'$ as follows:

$$\begin{aligned} t &= \mathrm{f}u \\ &\equiv \mathrm{f}(v^C \parallel \mathrm{rl}_{\mathbf{nr}} v^{T-C}) \\ &\equiv \mathrm{f}(v^C \parallel \mathrm{rl}_{\mathbf{nr}}\mathrm{f}v') \\ &\equiv \mathrm{f}(\mathrm{rl}_{\mathbf{nr}}\mathrm{f}v^C \parallel v') && \text{(A5)} \\ &= \mathrm{f}u' \end{aligned}$$

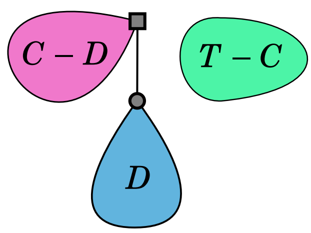
We now assume that $t = \mathrm{f}u$ with u's root in the same component as x. We reason by induction on the length of the unique path from the root r of $\mathcal{G}(u)$ to x in T. If it is zero then $x = r$ and we take $t^x = u$. Otherwise, with C the component of T common to x and r, x must be in a component D of $C - r$. Let r' be the unique neighbour of r in D. The length of the unique path from r' to x in T is one less than that from r to x.

By Lemma 2.15.2 there exists terms u^C and u^{T-C}, parsings of C and $T - C$ respectively, such that $u \equiv u^C \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}$.

By Lemma 2.15.1 there exists parsings v^{C-D} and v^D of $C - D$ (with root r) and D (with root r'), respectively, such that,

$$u^C \equiv v^{C-D} \parallel \mathrm{rl}_{\mathbf{r}} v^D.$$

In other words, $\mathcal{G}(u)$ has shape:



We conclude by induction hypothesis on the following parsing of T with root r':

$$t^{r'} = v^D \parallel \mathrm{rl}_{\mathbf{r}}(v^{C-D} \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}).$$

Indeed:

$$\begin{aligned}
t &= \mathrm{f}u \\
&\equiv \mathrm{f}(u^C \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}) \\
&\equiv \mathrm{f}((v^{C-D} \parallel \mathrm{rl}_{\mathbf{r}} v^D) \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}) \\
&\equiv \mathrm{f}(\mathrm{rl}_{\mathbf{r}} v^D \parallel (v^{C-D} \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C})) && \text{(A1,2)} \\
&\equiv \mathrm{f}(v^D \parallel \mathrm{rl}_{\mathbf{r}}(v^{C-D} \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C})) && \text{(A4)} \\
&= \mathrm{f}t^{r'}
\end{aligned}$$

□

Proposition 2.15.4 (Completeness of $\mathrm{Ax}_{\text{forest}}$)**.** *For every terms $t, u \in \mathcal{T}_{\Sigma_{\text{forest}}}$,*

$$t \equiv u \Leftarrow \mathcal{G}(t) \simeq \mathcal{G}(u).$$

Proof. Assume $\mathcal{G}(t)$ is isomorphic to $\mathcal{G}(u)$, and write T for the corresponding forest. We reason by induction on $|T|$.

We distinguish several cases depending on the structure of T:

- T **is empty;** we can only have $t = \emptyset = u$.

  We now assume that T is non-empty.

- T **is non-rooted;** necessarily, as it is non-empty, there must be rooted terms t' and u' such that $t = \mathrm{f}t'$ and $u = \mathrm{f}u'$. If t' and u' have different roots, we let r be t''s root, and by Lemma 2.15.3 we let u^r be a parsing of $\mathcal{G}(t')$ such that $u \equiv \mathrm{f}u^r$.

  Hence we can always assume that t' and u' share the same root. We deduce $t \equiv u$ by proving $t' \equiv u'$, as shown next.

- T **is rooted;** Assume that there is at least one component not containing the source in T, and let C be the component of T containing the source. By Lemma 2.15.2 there exists terms t^{T-C}, t^C, u^{T-C}, and u^C, respectively parsings of $T-C$, C, $T-C$, and C, such that $t \equiv t^C \parallel \mathrm{rl}_{\mathbf{nr}} t^{T-C}$ and $u \equiv u^C \parallel \mathrm{rl}_{\mathbf{nr}} u^{T-C}$.

  By induction hypothesis, $t^{T-C} \equiv u^{T-C}$ and $t^C \equiv u^C$; hence, $t \equiv u$.

  We now assume that T is connected: it is a rooted tree.

  If there exists at least two components in $T-r$ then let C be one of them. By Lemma 2.15.1 there exists terms t^{T-C} and u^{T-C} parsings of $T-C$, as well as terms t^C and u^C, parsings of C (with r' the unique neighbour of r in C as the root), such that $t \equiv t^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} t^C$ and $u \equiv u^{T-C} \parallel \mathrm{rl}_{\mathbf{r}} u^C$.

By induction hypothesis we have that $t^{T-C} \equiv u^{T-C}$ and $t^C \equiv u^C$ so that $t \equiv u$.

We now assume that there is precisely one component in $T - r$. Up-to removing parallel compositions with terms reduced to their roots (via A3), we can suppose that $t \equiv \mathrm{rl}_{\mathbf{r}} t'$ and $u \equiv \mathrm{rl}_{\mathbf{r}} u'$ where both t' and u' are parsings of $T - r$ in which the unique neighbour of r is the root. By induction hypothesis $t' \equiv u'$ so that $t \equiv u$.

The final case occurs when T is reduced to its root. Again, up-to removing parallel compositions with terms reduced to their roots using A3, we can assume that $t \equiv \mathrm{rl}_{\mathbf{nr}} \emptyset$ and $u \equiv \mathrm{rl}_{\mathbf{nr}} \emptyset$. □

2.16 A finite equational axiomatisation of forests. As a consequence of Proposition 2.14.3 and 2.15.4, we get:

Theorem 2.16.1. *For every terms $t, u \in \mathcal{T}_{\Sigma_{\text{forest}}}$,*

$$t \equiv u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u).$$

3
Graph shapes

3.1 Introduction. We can understand the terms in $\mathcal{T}_{\Sigma_{\text{graph}}}(\mathcal{V})$ as "shapes" in which variables represent all possible choices of terms for substitution. In the first section of this chapter, we introduce graph shapes (§3.2 to §3.6). Graphs of arity k can be substituted in place of edges of the same arity. Hence we introduce a notion of hypergraphs whose edges are labelled by variables.

In the second section of this chapter, we introduce a variation of Σ_{graph} whose terms are interpreted as hypergraphs (§3.7 to §3.9). Then, we state the main problem this thesis is concerned with: the existence, for every non-negative integer k, of a finite equational axiomatisation for graphs of tree-width at most k (§3.10 to §3.13).

Finally, in a last section, we provide a positive answer for the set of all graphs (§3.14 to §3.16).

GRAPH SHAPES, CONTEXTS AND SUBSTITUTIONS

3.2 Hypergraphs. Given a set A, we write $A^{\neq}$ to refer to the ranked set of duplicate-free lists over A, where the arity of a list is its length. Henceforth, we assume a ranked set $\mathcal{V}$ of variables.

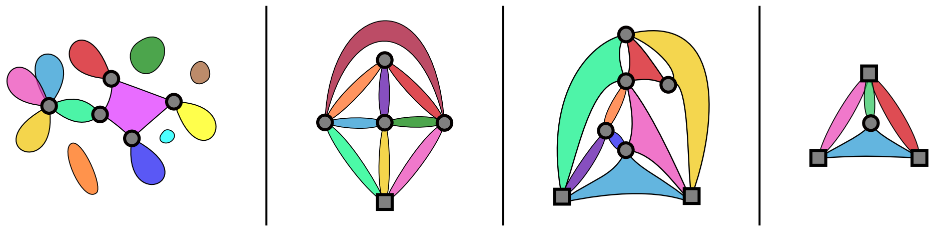


Figure 3.1: This figure depicts four graphs. Edge labels and orientations, as well as source orderings are typically not shown. Edges are represented by coloured areas connecting the vertices the edge is incident with.

Definition 3.2.1. A *(labelled and ordered sourced) hypergraph* is a tuple $\langle V, I, E, i, l\rangle$ where:

- V is a finite set of *vertices*,
- $I \in V^{\neq}$ is the *interface*,
- E is a finite ranked set of *(hyper)edges*,
- $i\colon E \to V^{\neq}$ is an arity-preserving function mapping an edge to its *incident vertices*, and,
- $l\colon E \to \mathcal{V}$ is an arity-preserving function giving the *label* of each edge.

Henceforth, the term "graph" is often used to refer to hypergraphs. When a confusion with simple graphs is possible, we retain the names "hypergraph" and "simple graph".

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)*. The *cardinality* of a graph G is its number of elements. It is denoted by $\#G = |G| + \|G\|$. The *arity* of a graph G is that of its interface, i.e. its number of sources. It is denoted by $\mathrm{ar}(G)$. Edge labels, i.e. variables in $\mathcal{V}$, are typically denoted by $a, b, c, \ldots$.

In this thesis, all concrete hypergraphs we consider have edges with pairwise distinct labels. Hence, we often use its label to refer to an edge.

Figure 3.1 depicts several graphs, and details graph depiction conventions.

3.3 Generalising graph terminology. The *skeleton* of a graph G is the simple graph whose vertex set is that of G, and in which two vertices are adjacent when they are either both sources or incident to a common edge in G.

In other words, the skeleton of a hypergraph is obtained by turning all its hyperedges as well as its interface into simple cliques.

Most definitions from Chapter 1 have natural generalisations to hypergraphs.

For example, two vertices of a hypergraph are called *adjacent* when they are incident to a common edge. Observe that the adjacency relation on a hypergraph is distinct from the adjacency relation of its skeleton. Indeed, two non-adjacent sources become adjacent in the skeleton.

In the current paragraph as well as in the next two, we detail some definitions whose generalisation from simple graphs to hypergraphs require particular attention.

A hypergraph is:

- *empty* if its only elements are its sources. The empty hypergraph of arity k is denoted by $\emptyset_k$;
- *atomic* if its only inner element is an edge, whose incident vertices are all the sources. If, moreover, as lists of vertices, the incident vertices of the edge and the interface of the graph coincide, and if the edge is labelled by $a \in \mathcal{V}$, then we write a to refer to this atomic graph.

Empty and atomic graphs are called *trivial hypergraphs*.

In hypergraphs, *walks* are sequences of elements such that any two consecutive elements consist of an edge and of one of its incident vertices (possibly in reversed order). *Paths* are *walks* in which vertices are not repeated. An *inner walk* is a walk made only of inner elements, except possibly for the endpoints.

Observe that unlike in simple graphs, walks in hypergraphs may begin and end on edges.

Definition 3.3.1. Two graphs $G = (V_G, I_G, E_G, i_G, l_G)$ and $H = (V_H, I_H, E_H, i_H, l_H)$ are called *isomorphic* if there exists two bijections $\varphi\colon V_G \to V_H$ and $\psi\colon E_G \to E_H$ such that

- $i_H \circ \psi = \varphi^{\neq} \circ i_G$ (preservation of incident vertices),
- $l_H \circ \psi = l_G$ (preservation of labels), and,
- $I_H = \varphi^{\neq}(I_G)$ (preservation of the interface),

where $\varphi^{\neq}$ denotes the point-wise extension of φ to duplicate-free lists.

3.4 Treewidth of a graph.

Definition 3.4.1. A *tree decomposition* of a graph is a tree whose nodes are labelled by sets of vertices, called *bags*, such that

- every vertex of the graph appears in a bag,
- for each pair of vertices incident to a common edge, there is a bag containing both of them,
- there is a bag containing all sources,
- if a given vertex appears in two bags b and b' of the tree decomposition, then all bags along the unique path from b to b' contain it too.

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.

In particular, the treewidth of a graph is at least its arity minus one.
Tree decomposition of a hypergraph and of its skeleton coincide, proving:

Proposition 3.4.2. *The treewidth of a graph and of its skeleton are equal.*

3.5 Graph substitutions and contexts. For each non-negative integer k, we assume the existence of a label $[]_k \in \mathcal{V}$ of arity k we refer to as the *hole (label)* of arity k. A *(graph) substitution* is an arity-preserving total function from variables to hypergraphs which is the identity on holes and without holes in the images of non-hole labels.

As for term substitutions, a graph substitution σ can be extended from variables to a mapping $\hat{\sigma}$ on graphs: $\hat{\sigma}(G)$ is the graph obtained from G by replacing all its edges by copies of the graphs in the image of σ.

More precisely, given an edge e of G, labelled by a, and of arity n, if $x_1, \ldots, x_n$ are the vertices incident to e in G, and $r_1, \ldots, r_n$ are the sources of the atomic graph labelled by a, then the *substitution of e along σ* in G is:

$$G[\sigma(a)/e] = ((G - e) \uplus \sigma(a))/x_1 r_1/\ldots/x_n r_n.$$

where the resulting interface is that of G.

Observe that this definition is similar to that of parallel compositions from §1.16.

Substitutions are illustrated in Figure 3.2.

A *graph context* C is a hypergraph in which exactly one edge is labelled by a hole label. The *substitution of the hole* in C by a graph G (of the same arity than the hole) is denoted by $C[G]$.

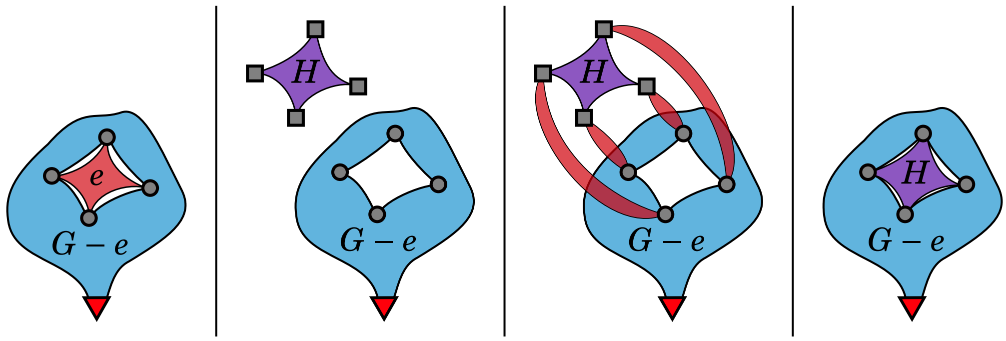


Figure 3.2: Illustration of the substitution by a graph H of an edge e of arity four in a graph G. From left to right: G, the disjoint union of $G-e$ and H, the addition of the edges $x_i r_i$, and their contractions. The red triangular source of G represents all its sources.

By combining isomorphisms for two graphs and the images of two substitutions, we obtain:

Proposition 3.5.1. *For all graphs G, H and all substitutions σ_1, σ_2, if $G \simeq H$ and $\sigma_1(a) \simeq \sigma_2(a)$ for all variables a, then $\hat{\sigma_1}(G) \simeq \hat{\sigma_2}(H)$.*

Treewidth is also preserved by graph substitutions:

Proposition 3.5.2. *If a graph G and the graphs in the image of a substitution σ all have treewidth at most k, then so does $\hat{\sigma}(G)$.*

Proof. Starting from a tree decomposition T for G, we obtain a tree decomposition for $\hat{\sigma}(G)$ by adding, for each a-labelled edge e of G, a copy T_e of a tree decomposition for $\sigma(a)$, and an edge connecting a bag of T containing the incident vertices of e to a bag of T_e containing the sources of $\sigma(a)$. This construction does not create bags, and thus preserves their maximal size. □

3.6 Graph shapes. Given two graphs G and H, we say that G *has shape* H when $G \simeq \hat{\sigma}(H)$ for some substitution σ. We also say that G has an H-*shape*.

Examples of graph shapes are given in Figure 3.3.

A SIGNATURE FOR GRAPHS OF BOUNDED TREEWIDTH

3.7 Footprint of a graph. The *footprint* of a hypergraph G is the sourced simple graph obtained by replacing each edge by a clique, removing multi-edges when necessary.

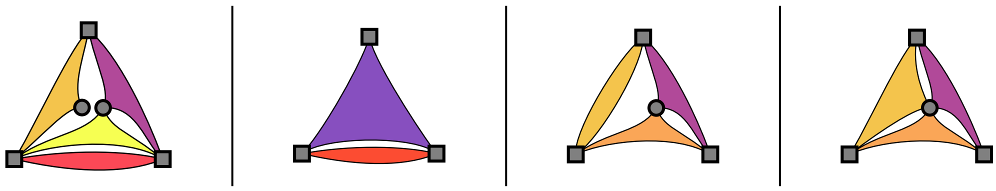


Figure 3.3: Four shapes of the leftmost graph (including itself).

Hence the footprint of a graph is in-between the graph and its skeleton. More precisely the skeleton of the footprint is the skeleton of the graph.

3.8 Treewidth via graph operations. The graph operations defined in §1.16 on sourced simple graphs generalise to graphs:

Definition 3.8.1. 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 identifying 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.

The footprint of a parallel composition is the parallel composition of the footprints. Similar statements hold for each of these operations.

As a consequence, by Proposition 3.4.2, the statement of Proposition 1.16.3 applies to graphs:

Proposition 3.8.2. *Let $k \geq 1$ be an integer. The closure of the empty graph and of the edges of arity at most $k+1$ under parallel compositions, forgets, lifts (limiting the arity to $k+1$), and permutations is exactly the set of hypergraphs of treewidth at most k.*

3.9 Interpreting Σ_{graph} on graphs. Since variables of sort two are interpreted as edges incident to two vertices, we do not need the symbol e in Σ_{graph} and we use instead the following signature:

$$\Sigma_{\text{graph}} = \{ \quad \begin{aligned} \parallel &: k^2 \to k \\ \mathrm{l} &: k \to k+1 \\ \mathrm{f} &: k+1 \to k \\ p &: k \to k \\ \emptyset_0 &: 0 \quad \}. \end{aligned}$$

In Backus-Naur form:

$$\begin{aligned} t_0 &= \qquad \mathrm{f}t_1 \;\;\mid pt_0 \;\;\mid t_0 \parallel t_0 \;\;\mid \emptyset_0 \\ t_k &= \mathrm{l}t_{k-1} \mid \mathrm{f}t_{k+1} \mid pt_k \;\;\mid t_k \parallel t_k. \end{aligned}$$

A variable a of arity k is interpreted as the atomic graph of arity k whose sole edge is labelled a; function symbols are interpreted using the graph operations from Definition 3.8.1. This provides a function $\mathcal{G}$ from terms of $\mathcal{T}_{\Sigma_{\text{graph}}}(\mathcal{V})$ to graphs.

Observe that the notations used in Chapter 2 for variables $(x, y, z, \dots)$, are reserved for vertices here, and variables are denoted by $a, b, c, \dots$ instead, avoiding clashes.

As before we say that a term t *denotes* a graph G, or that t is a *parsing* of G, when $\mathcal{G}(t) \simeq G$, and the *width* of a term is the maximal arity of its subterms, minus one.

The subsignature of Σ_{graph} containing the symbols of arity at most $k+1$ is denoted by Σ^k_{graph}.

Henceforth, we write $\mathcal{T}(\mathcal{V})$ and $\mathcal{T}_k(\mathcal{V})$ for $\mathcal{T}_{\Sigma_{\text{graph}}}(\mathcal{V})$ and $\mathcal{T}_{\Sigma^k_{\text{graph}}}(\mathcal{V})$, respectively.

As a consequence of Proposition 3.8.2 we get:

Corollary 3.9.1. *A graph has treewidth at most k if and only if it has a parsing in $\mathcal{T}_k(\mathcal{V})$.*

We can then make the link between contexts and substitutions of terms and of graphs:

Proposition 3.9.2. *A term t is a term context if and only if the graph $\mathcal{G}(t)$ is a graph context.*

In this case, given a term u, $\mathcal{G}(t[u]) \simeq \mathcal{G}(t)[\mathcal{G}(u)]$.

A function σ from variables to terms is a term substitution if and only if the mapping $\mathcal{G}(\sigma)\colon a \mapsto \mathcal{G}(\sigma(a))$ is a graph substitution. In this case, if t is a term $\mathcal{G}(\hat{\sigma}(t)) \simeq \widehat{\mathcal{G}(\sigma)}(\mathcal{G}(t))$.

3.10 Graph isomorphism as an equational theory. Let k be a non-negative integer. Write $\cong$ for the relation defined on terms of $\mathcal{T}(\mathcal{V})$ by $u \cong v$ if and only if $\mathcal{G}(u) \simeq \mathcal{G}(v)$, and write $\cong_k$ for the restriction of $\cong$ to terms of width at most k.

Propositions 3.9.2 and 3.5.1 show that $\cong$ and $\cong_k$ are equational theories.

3.11 A finite equational axiomatisation of graphs?

Problem 3.11.1. Is there a finite set of axioms on $\mathcal{T}(\mathcal{V})$ whose associated equational theory is $\cong$?

On general graphs, arity is unbounded. Simple properties such as the commutativity of the parallel composition must be expressible at each arity. Hence, "finite" refers to a finite set of axioms at each arity.

We answer this problem positively in the next section.

3.12 The treewidth axiomatisation conjecture. We now introduce, in the form of a conjecture, the main problem we are concerned with:

Conjecture 3.12.1 (Treewidth axiomatisation conjecture)**.** *Is there a finite set of axioms on $\mathcal{T}_k(\mathcal{V})$ whose associated equational theory is $\cong_k$?*

This conjecture has first been formulated by Courcelle and Engelfriet [12, p. 119] for a different syntax and in the form a problem.

We solved the case of $k = 1$ in §2.15), albeit for a different signature. A positive answer is provided in §4.9, §4.14, and §4.20 for $k = 0$, 1, and 2, respectively, and Chapter 5 is devoted to the case of $k = 3$.

3.13 Soundness and completeness. Let Σ be a signature among Σ_{graph} and Σ^k_{graph}. Let $\equiv_{\text{Ax}}$ be the equational theory associated to a set Ax of axioms. The set Ax is called:

- *sound* when, for every terms $t, u \in \mathcal{T}(\mathcal{V})$, we have:

$$t \equiv_{\text{Ax}} u \Rightarrow \mathcal{G}(t) \simeq \mathcal{G}(u),$$

- *complete* when, for every terms $t, u \in \mathcal{T}(\mathcal{V})$, we have:

$$t \equiv_{\text{Ax}} u \Leftarrow \mathcal{G}(t) \simeq \mathcal{G}(u).$$

To answer Problem 3.11.1 or an instance of the treewidth axiomatisation conjecture, we typically prove soundness and then completeness, the latter being the part where most difficulties arise.

AN AXIOMATISATION OF ALL GRAPHS

3.14 Soundness made easy. Let Ax be a set of *axioms* on terms over either Σ_{graph} or Σ^k_{graph}. Write $\equiv_{\text{Ax}}$ for the associated equational theory.

Proposition 3.14.1. *If, for every equation* $(t = u) \in \text{Ax}$, *we have* $\mathcal{G}(t) \simeq \mathcal{G}(u)$, *then* Ax *is sound.*

Proof. Since $\cong$ and $\cong_k$ are equational theories containing Ax (by hypothesis), they contain the least equational theories containing Ax for their respective signature, that is, they contain $\equiv_{\text{Ax}}$. □

3.15 Easy axioms. In the next definition, we give a list of axioms corresponding to simple properties of the graphs operations in Σ_{graph}. Each equation is an *axiom scheme*, that is, a uniform representation of several axioms, each axiom being an instance of the scheme at an arity compatible with the signature under study (Σ_{graph} of Σ^k_{graph}).

For example, the associativity of the parallel operation holds at each arity, but with Σ^k_{graph}, the corresponding scheme corresponds to $k+1$ axioms, since the maximum arity is $k+1$.

Moreover, the variables appearing in the axiom schemes may not have the same arities; a and b have the same arity in $a \parallel b = b \parallel a$ but not in $\mathrm{f}a \parallel b = \mathrm{f}(a \parallel \mathrm{l}b)$ where $\mathrm{ar}(b) = \mathrm{ar}(a) + 1$.

Definition 3.15.1. We write $\equiv$ to refer to the least equational theory on $\mathcal{T}(\mathcal{V})$ containing the following axiom schemes, and C for the corresponding set of axioms; let a, b, c be variables, p and q permutations, r the permutation swapping the last two elements, and $\dot{p}$ the extension of p from $[1, k]$ to $[1, k+1]$ fixing the last element:

A1. $a \parallel (b \parallel c) = (a \parallel b) \parallel c$,

A2. $a \parallel b = b \parallel a$,

A3. $a \parallel \emptyset_k = a$,

A4. $pqa = (p \circ q)a$,

A5. $\mathrm{id}a = a$ where id denotes the identity permutation,

A6. $p(a \parallel b) = pa \parallel pb$,

A7. $p\emptyset_k = \emptyset_k$,

A8. $\mathrm{l}(a \parallel b) = \mathrm{l}a \parallel \mathrm{l}b$,

A9. $p\mathrm{f}a = \mathrm{f}\dot{p}a$,

A10. $\mathrm{l}pa = \dot{p}\mathrm{l}a$,

A11. $\mathrm{lf}a = \mathrm{f}r\mathrm{l}a$,

A12. $\mathrm{ll}a = r\mathrm{ll}a$,

A13. $\mathrm{f}a \parallel b = \mathrm{f}(a \parallel \mathrm{l}b)$,

FA. $\mathrm{ff}a = \mathrm{ff}ra$.

We write E for C\{FA}.

Each of these axioms correspond to a graph, some of which are illustrated in Figure 3.4.

The notations C and E stand for "Clique Axioms" and "Easy Axioms", respectively. Clique axioms are related to a class of vertices called *clique points* which we introduce in the next chapter.

The axiom scheme FA (for "Forget Atomis"), is the first of a series of important axioms we call forget axioms.

A *forget axiom* is an axiom of the form $\mathrm{f}t = \mathrm{f}u$ where $\mathrm{f}t$ and $\mathrm{f}u$ denote the same graph, but the two forget operations correspond to distinct vertices in that graph.

The axiom FA is an example of forget axiom. The underlying graph is obtained from an atomic graph by forgetting the last two sources (see Figure 3.4).

To obtain a finite equational axiomatisation of all graphs, FA is the only forget axiom we need. In the next two chapters, forget axioms are crucial to obtain finite equational axiomatisation at bounded treewidth. Discovering new forget axioms for graphs of treewidth at most three is a key contribution of our work.

Proposition 3.15.2 (Soundness)**.** *If $t \equiv u$ then $t \cong u$.*

Proof sketch. By Proposition 3.14.1 it suffices to verify the two sides of each axiom denote the same graphs. Those are shown in Figure 3.4 for some of the axioms. □

The axioms of Definition 3.15.1 intuitively correspond to those given by Courcelle and Engelfriet [12, p.117], modulo some translation since signatures and graph operations differ.

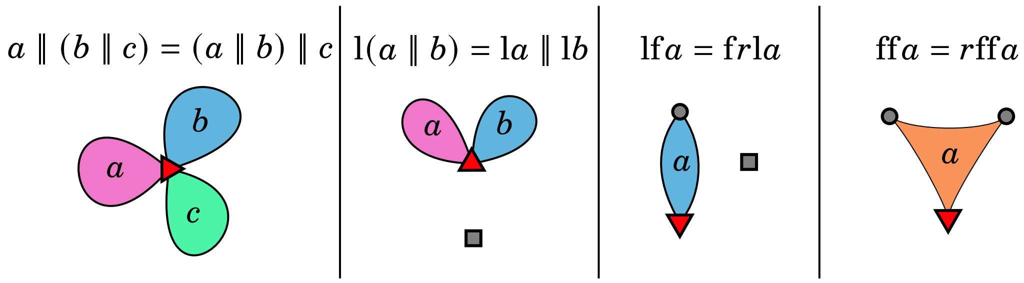


Figure 3.4: The graphs corresponding to the following axioms: A1, A8, A11, and FA. The red triangular sources represent sources which are not modified by the operations appearing in the axioms.

3.16 A "finite" equational axiomatisation of all graphs. We show how C provides a finite equational axiomatisation of the graph isomorphism, answering positively Problem 3.11.1.

Let t be a parsing of a graph G. Write $n = |G| - \mathrm{ar}(G)$ for the number of inner vertices of G, $a_1, \ldots, a_m$ for the $m = \|G\|$ labels of G's edges, and $n_i = |G| - \mathrm{ar}(a_i)$ for the number of vertices of G which are not incident to a_i ($i = 1, \ldots, m$).

We say that a term t is *canonical* whenever either $m > 1$ and there exists permutations $p_1, \ldots, p_m$ such that:

$$t = \mathrm{f}^n(p_1 \mathrm{l}^{n_1} a_1 \parallel \ldots \parallel p_m \mathrm{l}^{n_m} a_m)$$

or $m = 0$ and:

$$t = \mathrm{f}^n \emptyset_{|G|}$$

Proposition 3.16.1. *Each graph admits a canonical parsing.*

Proof. Let G be a graph. Write $n = |G| - \mathrm{ar}(G)$ for the number of inner vertices of G.

If G contains no edges, then $\mathrm{f}^n \emptyset_{|G|}$ is a canonical parsing of G.

Otherwise, fix a total order $\prec$ on the vertices of the graph G such that the ith source of G is the ith vertex of the order $\prec$.

Assume a total order on the edges of G and denote by a_i the label of the ith edge of G along this order, and by $n_i = |G| - \mathrm{ar}(a_i)$ the number of vertices of G which are not incident to the ith edge of G along this order.

We construct a canonical parsing via the following steps:

- for every $i \in [1, \ldots, \|G\|]$, lift a_i exactly the number of times required to add the missing vertices: $\mathrm{l}^{n_i} a_i$,

- permute the result to follow the order $\prec$; let p_i be a permutation such that for $1 \leq j \leq \mathrm{ar}(a_i)$ and $1 \leq k \leq |G|$, if the jth vertex of $\mathcal{G}(a_i)$ is the kth vertex of $\prec$, then $p_i(j) = k$.

  We consider the term $p_i \mathrm{l}^{n_i} a_i$,

- put the results in parallel and forget all n non-source vertices of G:

$$\mathrm{f}^n (p_1 \mathrm{l}^{n_1} a_1 \parallel \ldots \parallel p_{\|G\|} \mathrm{l}^{n_{\|G\|}} a_{\|G\|})$$

  is a canonical parsing of G. □

This result generalises to terms, by using axioms:

Proposition 3.16.2. *For every term $t \in \mathcal{T}(\mathcal{V})$, there exists a canonical term u such that $t \equiv u$.*

Proof. Denote by G the graph $\mathcal{G}(t)$. We reason by induction on the lexicographic product of $\#G$ with $n = |G| - \mathrm{ar}(G)$, and with the *size* of t. The value n is the number of inner vertices of G.

The *size* $|t|$ of a term is defined by induction as follows:

- 0 if t is a constant,
- $1 + |t_1| + \cdots + |t_m|$ if $t = f(t_1, \ldots, t_m)$.

We perform case analysis on the possible definitions of t:

- $t = \emptyset_{\mathrm{ar}(G)}$; t is canonical.
- $t = pt'$; by induction hypothesis there exists a canonical term of shape $\mathrm{f}^n u$ such that $t' \equiv \mathrm{f}^n u$.

  We consider the possible values of n.

  <u>$n > 0$;</u> using A9 we have

$$t \equiv \mathrm{f}\dot{p}\mathrm{f}^{n-1}u.$$

  By induction hypothesis there exists a canonical term v such that:

$$\dot{p}\mathrm{f}^{n-1}u \equiv v.$$

  So $t \equiv \mathrm{f}v$ which is canonical.

  <u>$n = 0$, i.e., $|G| = \mathrm{ar}(G)$;</u> in particular, $t' \equiv u$.

  If G contains no edges then $u = \emptyset_{|G|}$ and using A7 we have $t \equiv \emptyset_{|G|}$ which is canonical.

Otherwise, G contains at least one edge, and

$$u = p_1 \mathrm{l}^{n_1} a_1 \parallel \ldots \parallel p_{\|G\|} \mathrm{l}^{n_{\|G\|}} a_{\|G\|}.$$

By applying A6 and then A4 repeatedly, we have

$$t \equiv (p \circ p_1) \mathrm{l}^{n_1} a_1 \parallel \ldots \parallel (p \circ p_{\|G\|}) \mathrm{l}^{n_{\|G\|}} a_{\|G\|}$$

which is canonical.

- $t = \mathrm{f}t'$; by induction hypothesis there exists a canonical term u such that $t' \equiv u$. Then $t \equiv \mathrm{f}u$ which is canonical.

- $t = \mathrm{l}t'$; by induction hypothesis there exists a canonical term $\mathrm{f}^n u$ such that $t' \equiv \mathrm{f}^n u$.

  We perform case analysis on the possible values of n.

  $\underline{n > 0;}$ using A11 we have

  $$t \equiv \mathrm{f} r \mathrm{l} \mathrm{f}^{|G|-1-\mathrm{ar}(G)} u.$$

  By induction hypothesis there exists a canonical term v such that:

  $$r \mathrm{l} \mathrm{f}^{n-1-1\mathrm{ar}(G)} u \equiv v.$$

  So $t \equiv \mathrm{f}v$ which is canonical.

  $\underline{n = 0, \text{ i.e., } |G| = \mathrm{ar}(G)}$; in particular $t' \equiv u$.

  If G contains no edges, then $u = \emptyset_{|G|-1}$, and we have $t \equiv \emptyset_{|G|}$, which is canonical.

  Otherwise G contains at least one edge and

  $$u = p_1 \mathrm{l}^{n_1 - 1} a_1 \parallel \ldots \parallel p_{\|G\|} \mathrm{l}^{n_{\|G\|} - 1} a_{\|G\|}.$$

  Using A8 and A10 repeatedly, we have

  $$t \equiv \dot{p}_1 \mathrm{l}^{n_1} a_1 \parallel \ldots \parallel \dot{p}_{\|G\|} \mathrm{l}^{n_{\|G\|}} a_{\|G\|}$$

  which is canonical.

- $t = u \parallel v$; denote $|\mathcal{G}(u)|$ by $|u|$ and $|u| - \mathrm{ar}(G)$ by n_u, and assume similar notations for v. By induction hypothesis there exists canonical terms $\mathrm{f}^{n_u} u'$ and $\mathrm{f}^{n_v} v'$ such that $u \equiv \mathrm{f}^{n_u} u'$ and $v \equiv \mathrm{f}^{n_v} v'$.

  We perform case analysis on the possible values of $n_u > 0$.

$\underline{n_u > 0}$; using A13, we have

$$t \equiv \mathrm{f}(\mathrm{f}^{n_u-1}u' \parallel \mathrm{l}v)$$

We conclude by induction hypothesis on $\mathrm{f}^{n_u-1}u' \parallel \mathrm{l}v$.

$\underline{n_u = 0}$; in particular, $u \equiv u'$.

If $\mathcal{G}(u)$ contains no edges, then $u' = \emptyset_{|u|}$ and by A3 we have $t \equiv \mathrm{f}^{n_v}v'$ which is canonical.

Thanks to A2, we can perform a similar analysis on v, and assume $n_v = 0$ and that $\mathcal{G}(v)$ contains at least one edge. Then $t \equiv u' \parallel v'$ which is canonical. □

At this point, we have no control on the specific canonical parsing we get from Proposition 3.16.2. Proposition 3.16.5 below states that canonical parsings are related by $\equiv$. To prove this proposition we introduce the next two lemmas:

Lemma 3.16.3. *Let a be a variable of arity n and $k \leq n$ a non-negative integer. For every permutation p on $[1,\ldots,n]$ that fixes the first $n-k$ integers, we have:*

$$\mathrm{f}^k pa \equiv \mathrm{f}^k a.$$

Proof. We reason by induction on k.

If p is the identity permutation then we conclude using A5.

This is in particular the case when $k \leq 1$.

Assume that $k \geq 2$ and that p is not the identity. With s the permutation swapping n with $p^{-1}(n)$, we have $p \circ s(n) = p(p^{-1}(n)) = n$. So $p \circ s$ can be written as the extension $\dot{q}$ of a permutation on $[1,\ldots,n-1]$ to $[1,\ldots,n]$ by fixing n. We have $p = p \circ (s \circ s) = \dot{q} \circ s$, and:

$$\begin{aligned}
\mathrm{f}^k pa &= \mathrm{f}^k(\dot{q} \circ s)a \\
&\equiv \mathrm{f}^k \dot{q}sa && \text{(A4)} \\
&\equiv \mathrm{f}^{k-1} q\mathrm{f}sa && \text{(A9)} \\
&\equiv \mathrm{f}^k sa && \text{(by induction hypothesis)}
\end{aligned}$$

Let r be the permutation swapping n with $n-1$ and s' the permutation on $[1,\ldots,n-1]$ swapping $n-1$ with $p^{-1}(n)$. We have $s = r \circ \dot{s'} \circ r$, and:

$$
\begin{aligned}
\mathrm{f}^k pa &\equiv \mathrm{f}^k (r \circ \dot{s'} \circ r)a \\
&\equiv \mathrm{f}^k r\dot{s'}ra && \text{(A4)} \\
&\equiv \mathrm{f}^k \dot{s'}ra && \text{(FA)} \\
&\equiv \mathrm{f}^{k-1} s'\mathrm{f}ra && \text{(A9)} \\
&\equiv \mathrm{f}^k ra && \text{(by induction hypothesis)} \\
&\equiv \mathrm{f}^k a. && \text{(FA)} \quad \square
\end{aligned}
$$

Lemma 3.16.4. *Let a be a variable of arity n, and k a non-negative integer. For every permutation p on $[1,\ldots,n+k]$ that fixes the first n elements, we have:*

$$p\mathrm{l}^k a \equiv \mathrm{l}^k a.$$

Proof. We reason by induction on k.

If p is the identity permutation then we conclude using A5.

This is in particular the case when $k \leq 1$.

Assume that $k \geq 2$ and that p is not the identity. With s the permutation swapping $n+k$ with $p(n+k)$, we have $s \circ p(n+k) = s(p(n+k)) = n+k$. So $s \circ p$ can be written as the extension $\dot{q}$ of a permutation on $[1,\ldots,n+k-1]$ to $[1,\ldots,n+k]$ by fixing $n+k$. We have $p = s \circ \dot{q}$.

,

$$
\begin{aligned}
p\mathrm{l}^k a &= (s \circ \dot{q})\mathrm{l}^k a \\
&\equiv s\dot{q}\mathrm{l}^k a && \text{(A4)} \\
&\equiv s\mathrm{l}q\mathrm{l}^{k-1} a && \text{(A10)} \\
&\equiv s\mathrm{l}^k a. && \text{(by induction hypothesis)}
\end{aligned}
$$

Let r be the permutation swapping $n+k$ with $n+k-1$ and s' the permutation on $[1,\ldots,n+k-1]$ swapping $n+k-1$ with $p(n+k)$. We have $s = r \circ \dot{s'} \circ r$.

$$
\begin{aligned}
p\mathrm{l}^k a &= (r \circ \dot{s'} \circ r)\mathrm{l}^k a \\
&\equiv r\dot{s'}r\mathrm{l}^k a && \text{(A4)} \\
&\equiv r\dot{s'}\mathrm{l}^k a && \text{(A12)} \\
&\equiv r\mathrm{l}s'\mathrm{l}^{k-1} a && \text{(A10)} \\
&\equiv r\mathrm{l}^k a && \text{(by induction hypothesis)} \\
&\equiv \mathrm{l}^k a && \text{(A12)} \quad \square
\end{aligned}
$$

Proposition 3.16.5. *For every pair of canonical terms t and u denoting the same graph, we have*

$$t \equiv u.$$

Proof. Note G for the graph t and u are parsings of. Write n for $|G| - \mathrm{ar}(G)$ the number of inner vertices of G.

By definition of canonical parsing, there are terms t' and u' of arity $|G|$ such that $t = \mathrm{f}^n t'$ and $u = \mathrm{f}^n u'$.

The graphs $\mathcal{G}(t')$ and $\mathcal{G}(u')$ share the same elements and edge incidences. The only possible difference is between the orderings of the two interfaces. The latter are duplicate-free lists of all of $|G|$'s vertices. Nonetheless, the ith source of G must remain the ith source in $\mathcal{G}(t')$ and in $\mathcal{G}(u')$.

Let p be a permutation on G's vertices such that applying p on the interface of $\mathcal{G}(t')$ gives the interface of $\mathcal{G}(u')$.

Using A5 and A4 gives

$$t \equiv \mathrm{f}^n p^{-1}(pt')$$

By Lemma 3.16.3 we have $t \equiv \mathrm{f}^n pt'$. By definition of p, $\mathcal{G}(pt') \simeq \mathcal{G}(u')$. By Proposition 3.16.2, there exists a canonical parsing t'' such that $pt' \equiv t''$.

If there are no edges in G then both t'' and u' must be $\emptyset_{|G|}$. Otherwise there is at least one edge in G and we have

$$t'' = p_1^t 1^{n_1^t} a_1^t \parallel \dots \parallel p_{\|G\|}^t 1^{n_{\|G\|}^t} a_{\|G\|}^t$$

and,

$$u' = p_1^u 1^{n_1^u} a_1^u \parallel \dots \parallel p_{\|G\|}^u 1^{n_{\|G\|}^u} a_{\|G\|}^u$$

with $n_i^t = |G| - \mathrm{ar}(a_i)^t$, $n_i^u = |G| - \mathrm{ar}(a_i)^u$, and for some permutations p_i^t and p_i^u, and variables a_i^t and a_i^u $(i = 1, \dots, \|G\|)$.

Using A1 and A2 (commutativity and associativity of the parallel composition), we can assume that the edges resulting from a_i^u and a_i^t in $\mathcal{G}(t)$ and $\mathcal{G}(u)$, respectively, are associated by an isomorphism between $\mathcal{G}(t)$ and $\mathcal{G}(u)$. Hence the variables a_i^t and a_i^u are equal and we denote them by a_i with $n_i = n_i^t = n_i^u$ $(i = 1, \dots, \|G\|)$:

$$t'' = p_1^t 1^{n_1} a_1 \parallel \dots \parallel p_{\|G\|}^t 1^{n_{\|G\|}} a_{\|G\|}$$

and

$$u' = p_1^u 1^{n_1} a_1 \parallel \dots \parallel p_{\|G\|}^u 1^{n_{\|G\|}} a_{\|G\|}.$$

To prove that $t \equiv u$, it remains to prove that

$$p_i^u 1^{n_i} a_i \equiv p_i^t 1^{n_i} a_i (i = 1, \dots, \|G\|).$$

As the associated graphs are isomorphic, p_i^u and p_i^t must differ on the last n_i elements only. Let q be a permutation such that $p_i^u = p_i^t \circ q$. By A4 and Lemma 3.16.4, we have $p_i^u 1^{n_i} a_i \equiv p_i^t 1^{n_i} a_i$. □

Theorem 3.16.6. *Given two terms t and u from $\mathcal{T}(\mathcal{V})$,*

$$t \equiv u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u)$$

Proof. Soundness is given by Proposition 3.15.2 and completeness is a direct consequence of Propositions 3.16.2 and 3.16.5. □

All axioms from Definition 3.15.1 have been used to prove this result.

Canonical parsings of graphs of small treewidth containing n vertices contain subterms of arity n. In contrast, the parsings of width at most k of a graph of treewidth at most k only contain subterms of arity at most $k + 1$. Relating the parsings of a graph using axioms with restrictions on the width of terms is much more involved; it is the topic of the next two chapters.

4
Decomposing bounded treewidth graphs

4.1 Introduction. Throughout this chapter we fix a non-negative integer k, and consider the terms in $\mathcal{T}_k(\mathcal{V})$. We denote by E_k and C_k the restrictions of the sets E and C of axioms, respectively, to terms of width at most k. Until §4.12, we write $\equiv_k$ for the equational theory on terms of width at most k induced by the equations of E_k.

In this chapter we provide a sequence of results towards a positive answer to the treewidth axiomatisation conjecture. In particular, assuming an axiom set containing C_k, we show that completeness can be reduced to the case of graphs we call *hard graphs*. This result resolves the treewidth axiomatisation conjecture when the treewidth is bounded by k with $k \leq 2$, and the case of $k = 3$ is solved in the next chapter.

Instead of outlining our results for each section as we do for the other chapters, we provide a summary of the main results we prove in in Figure 4.1.

Sets	Axioms	Technical results	Completeness	Incompleteness
E_k	A1-13	Full prime components Series decomposition Reaching some forget point	Treewidth at most 0	Treewidth at most $k \geq 1$
C_k	FA	Reaching clique points	Treewidth at most 1 The set of all graphs	Treewidth at most $k \geq 2$
A_k	FK	Reaching checkpoints and anchors	Treewidth at most 2	Treewidth at most $k \geq 3$
F_3	FX, FD	Hard graphs of treewidth three	Treewidth at most 3	

Figure 4.1: Summary of the technical, completeness, and incompleteness sequence of results we prove for the different axiom sets.

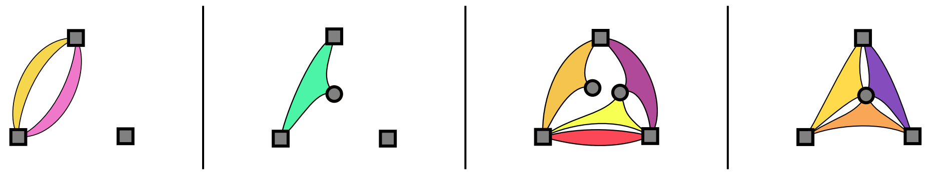


Figure 4.2: From the left-hand side to the right-hand side: the first graph is neither full nor prime, the second one is prime but not full, the third one is full but not prime, the last one is full prime.

The study of each graph notion is divided into two parts: one with results on graphs, and one demonstrating how these results translate to terms.

FULL PRIME COMPONENTS

4.2 Full prime components on graphs. We introduce a notion of connectivity specific to sourced graphs.

Definition 4.2.1. 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 if and only if all its elements are connected by inner paths.

Examples are shown in Figure 4.2

Proposition 4.2.2. *Every graph is isomorphic to a permutation of lifts of a full graph. This decomposition is unique up to the behaviour of the permutation on isolated sources.*

Proof. Starting from a graph G, build a full graph G' by deleting all isolated sources from G. The graph G can be obtained from G' by applying as many lifts as there are isolated sources in G, followed by a permutation to move the sources at their places in the interface of G. This permutation may behave in any way on the isolated sources. □

Proposition 4.2.3. *Every graph is isomorphic to a parallel composition of prime graphs. This decomposition is unique up to re-indexing of the parallel compositions.*

Proof. The relation C on inner elements such that xCy when x and y are connected via an inner path is an equivalence relation. Its equivalence classes seen as subgraphs provide the desired components. □

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 prime components, as shown in Proposition 4.2.2.

Proposition 4.2.4. *Full prime graphs of treewidth at most k and arity $k+1$ are atomic.*

Proof. Let G be a full prime graph of treewidth at most k and arity $k+1$.

For the sake of contradiction, assume G has an inner vertex x. Since G is full prime, x must be connected by inner paths to all elements of G. In G's skeleton, contract all the inner vertices onto x. This provides K_{k+2} as a minor, a contradiction by Proposition 1.15.6 as G has treewidth at most k.

Thus the only inner elements of G are edges; there must be exactly one since G is prime, and it must be incident to all sources since G is full. Therefore G is atomic. □

4.3 Full prime components on terms.

Proposition 4.3.1. *For every term $t \in \mathcal{T}_k(\mathcal{V})$, variable $a \in \mathcal{V}$, and graphs G and H of treewidth at most k, we have:*

1. *if $\mathcal{G}(t) \simeq \emptyset_i$ then $t \equiv_k \emptyset_i$ ($i \in [0, \dots, k+1]$);*
2. *if $\mathcal{G}(t) \simeq a$ then $t \equiv_k a$;*
3. *if $\mathcal{G}(t) \simeq pG$ then there is a parsing u of G such that $t \equiv_k pu$;*
4. *if $\mathcal{G}(t) \simeq \mathrm{l}G$ then there is a parsing u of G such that $t \equiv_k \mathrm{l}u$;*
5. *if $\mathcal{G}(t) \simeq G \parallel H$ then there are parsings u of G and v of H such that $t \equiv_k u \parallel v$.*

Proof. All items but the third one are proved by induction on t.

1. Since $\mathcal{G}(t)$ is empty, t cannot be a variable or a forget; the other cases are treated as follows:

   - $t = u \parallel v$; then $\mathcal{G}(u) \simeq \mathcal{G}(v) \simeq \emptyset_i$. We have:

$$\begin{aligned} t &= u \parallel v \\ &\equiv_k \emptyset_i \parallel \emptyset_i && \text{(induction hypothesis)} \\ &\equiv_k \emptyset_i && \text{(A3)} \end{aligned}$$

- $t = \mathrm{l}u$; then $\mathcal{G}(u) \simeq \emptyset_{i-1}$. We have:

$$\begin{aligned} t &= \mathrm{l}u \\ &\equiv_k \mathrm{l}\emptyset_{i-1} && \text{(induction hypothesis)} \\ &= \emptyset_i. \end{aligned}$$

- $t = pu$ for some permutation p; then $\mathcal{G}(u) \simeq \emptyset_i$. We have:

$$\begin{aligned} t &= pu \\ &\equiv_k p\emptyset_i && \text{(induction hypothesis)} \\ &\equiv_k \emptyset_i. && \text{(A7)} \end{aligned}$$

- $t = \emptyset_i$.

2. We prove the following stronger statement: for all permutations p, having $\mathcal{G}(t) \simeq pa$ implies $t \equiv_k pa$. The initial statement follows by choosing the identity permutation.

   Since $\mathcal{G}(t)$ is atomic, t cannot be $\emptyset_i$, a forget, or a lift; in the other cases we proceed as follows:

   - $t = u \parallel v$; by A2, without loss of generality, we can assume $\mathcal{G}(u) \simeq pa$ and $\mathcal{G}(v) \simeq \emptyset_n$ with $n = \mathrm{ar}(t)$. We have:

   $$\begin{aligned} t &= u \parallel v \\ &\equiv_k pa \parallel \emptyset_n && \text{(induction hypothesis and item 1. above)} \\ &\equiv_k pa. && \text{(A3)} \end{aligned}$$

   - $t = qu$ for a permutation q; then $\mathcal{G}(u) \simeq (q^{-1} \circ p)a$. We have:

   $$\begin{aligned} t &= qu \\ &\equiv_k q(q^{-1} \circ p)a && \text{(induction hypothesis)} \\ &\equiv_k pa. && \text{(A4)} \end{aligned}$$

   - $t = b$ for a variable b; the permutation p is the identity id and $b = a$. Then $t \equiv_k \mathrm{id}a$ using A5.

3. We take $u = p^{-1}t$ and use A5 and A4.

4. We prove the following stronger statement: for all permutations p and graphs G of treewidth at most k such that $\mathcal{G}(t) \simeq p\mathrm{l}G$, there exists a parsing u of G such that $t \equiv_k p\mathrm{l}u$; the initial statement follows by choosing the identity permutation.

   The term t cannot be a variable as its graph $\mathcal{G}(t)$ is not full. In the other cases we proceed as follows:

- $t = \emptyset_{i+1}$ for some integer $i \geq 0$; then we take $u = \emptyset_i$ and, using A7, we have $t \equiv_k p\mathrm{l}\emptyset_i$.
- $t = v \parallel w$; the isolated source s of $\mathcal{G}(t)$ corresponding to the lift operation in $p\mathrm{l}G$ remains isolated in $\mathcal{G}(v)$ and $\mathcal{G}(w)$. There exists graphs G_v and G_w such that $\mathcal{G}(v) \simeq p\mathrm{l}G_v$, $\mathcal{G}(w) \simeq p\mathrm{l}G_w$, and $G \simeq G_v \parallel G_w$. By induction hypothesis, there is a parsing v' of G_v such that $v \equiv_k p\mathrm{l}v'$. Similarly there is a parsing w' of G_w such that $w \equiv_k p\mathrm{l}w'$. We take $u = v' \parallel w'$:

$$\begin{aligned} t &= v \parallel w \\ &\equiv_k p\mathrm{l}v' \parallel p\mathrm{l}w' \\ &\equiv_k p\mathrm{l}(v' \parallel w') && \text{(A6 and A8)} \\ &= p\mathrm{l}u. \end{aligned}$$

- $t = qt'$ for some permutation q; then $\mathcal{G}(t') \simeq q^{-1}p\mathrm{l}G$. By induction hypothesis, we find a parsing u of G such that $t' \equiv_k (q^{-1} \circ p)\mathrm{l}u$. We deduce $t \equiv_k p\mathrm{l}u$ using A4.
- $t = \mathrm{f}t'$; then $\mathcal{G}(t) \simeq \mathrm{f}(\mathcal{G}(t')) \simeq p\mathrm{l}G$ so that, appending the source corresponding to the forget operation to G's interface gives a graph G' such that $G \simeq \mathrm{f}G'$ and $\mathcal{G}(t') \simeq \dot{p}r\mathrm{l}G'$, where r is the permutation that swaps the last two elements, and $\dot{p}$ is the extension of p fixing the last element (this can be proved via axioms, using A11 and A9). By induction hypothesis, we find a parsing u' of G' such that $t' \equiv_k (\dot{p} \circ r)\mathrm{l}u'$. We take $u = \mathrm{f}u'$:

$$\begin{aligned} t &= \mathrm{f}t' \\ &\equiv_k \mathrm{f}(\dot{p} \circ r)\mathrm{l}u' \\ &\equiv_k \mathrm{f}\dot{p}r\mathrm{l}u' && \text{(A4)} \\ &\equiv_k p\mathrm{f}r\mathrm{l}u' && \text{(A9)} \\ &\equiv_k p\mathrm{l}\mathrm{f}u' && \text{(A11)} \\ &= p\mathrm{l}u. \end{aligned}$$

- $t = \mathrm{l}t'$; then $\mathcal{G}(t) \simeq \mathrm{l}\mathcal{G}(t') \simeq p\mathrm{l}G$ and we distinguish two cases depending on whether p fixes the last element. Observe that p fixes the last element if and only if the two lift operations in $\mathrm{l}\mathcal{G}(t') \simeq p\mathrm{l}G$ concern the same source of G.
  - p fixes the last element; then $p = \dot{q}$ for some permutation q, and $\mathcal{G}(t') \simeq qG$.

By the item 3. above, we find a parsing u of G such that $t' \equiv_k qu$, and we have

$$\begin{aligned} t &= \mathrm{l}t' \\ &\equiv_k \mathrm{l}qu \\ &\equiv_k \dot{q}\mathrm{l}u && \text{(A10)} \\ &= p\mathrm{l}u. \end{aligned}$$

– p does not fix the last element; assume the last element is mapped to the ith one by p.
Recall that $t = \mathrm{l}t'$ and that $\mathcal{G}(t) \simeq \mathrm{l}\mathcal{G}(t') \simeq p\mathrm{l}G$.
Denote by s the source corresponding to the lift operation in $p\mathrm{l}G$. As p does not fix the last element, s is isolated in $\mathcal{G}(t')$. By Proposition 4.2.2, restricted to the source s only there is a permutation q and a graph G' such that $\mathcal{G}(t') \simeq q\mathrm{l}G'$, where the lift corresponds to s, and q maps the last element to the ith one. By induction hypothesis, there exists a parsing u' of G' such that $t' \equiv_k q\mathrm{l}u'$.
Let r be the permutation swapping the last two elements. The permutation $r^{-1} \circ \dot{q}^{-1} \circ p$ fixes the last element, so it is equal to $\dot{q}'$ for some permutation q'. By definition, we have $p = \dot{q} \circ r \circ \dot{q}'$.
We take $u = q'^{-1}\mathrm{l}u'$:

$$\begin{aligned} t &= \mathrm{l}t' \\ &\equiv_k \mathrm{l}q\mathrm{l}u \\ &\equiv_k \dot{q}\mathrm{l}\mathrm{l}u && \text{(A10)} \\ &\equiv_k \dot{q}r\mathrm{l}\mathrm{l}u && \text{(A12)} \\ &\equiv_k \dot{q}r\mathrm{l}(q' \circ q'^{-1})\mathrm{l}u && \text{(A5)} \\ &\equiv_k \dot{q}r\dot{q}'\mathrm{l}q'^{-1}\mathrm{l}u && \text{(A4 and A10)} \\ &= p\mathrm{l}q'^{-1}\mathrm{l}u. && \text{(A4)} \end{aligned}$$

5. We proceed as follows by induction on t:

- $t = \emptyset_n$ with $n = \mathrm{ar}(t)$; since $\mathcal{G}(t)$ is empty, we must have $G \simeq H \simeq \emptyset_n$. We take $u = v = \emptyset_n$ and by A3 we have $t \equiv_k u \parallel v$.
- $t = a$ for some variable a; since $\mathcal{G}(t)$ is atomic, w.l.o.g. we may assume that G is atomic and H empty. We take $u = a$ and $v = \emptyset_n$ with $n = \mathrm{ar}(t)$ and using A3 we have $t \equiv_k u \parallel v$.
- $t = \mathrm{l}t'$ for some term t'; the source s of $\mathcal{G}(t)$ corresponding to the lift operation in $\mathrm{l}t'$ is isolated. As $\mathcal{G}(t) \simeq G \parallel H$, s is isolated G and

H too. We have $\mathcal{G}(t') \simeq (G-s) \parallel (H-s)$. By induction hypothesis, there are parsings u' and v' of $G-s$ and $H-s$, respectively, such that $t' \equiv_k u' \parallel v'$.

We take $u = \mathrm{l}u'$ and $v = \mathrm{l}v'$, and via A8, we get $t \equiv_k \mathrm{l}(u' \parallel v') \equiv_k u \parallel v$.

- $t = pt'$ for some permutation p and term t'; we have $\mathcal{G}(t') \simeq p^{-1}G \parallel p^{-1}H$. By induction hypothesis, there exists parsings u' and v' of $p^{-1}G$ and $p^{-1}H$, respectively, such that $t' \equiv_k u' \parallel v'$.

  We take $u = pu'$ and $v = pv'$. Using A6 we get $t \equiv_k p(u' \parallel v') \equiv_k u \parallel v$.

- $t = t_1 \parallel t_2$; we split G and H accordingly: $G \simeq G_1 \parallel G_2$, and $H \simeq H_1 \parallel H_2$ such that $\mathcal{G}(t_1) \simeq G_1 \parallel H_1$ and $\mathcal{G}(t_2) \simeq G_2 \parallel H_2$; we use the induction hypothesis on t_1 and t_2 and we conclude using associativity and commutativity as expressed in A1 and A2.

- $t = \mathrm{f}t'$; then $\mathcal{G}(t) \simeq \mathrm{f}\mathcal{G}(t') \simeq G \parallel H$. We assume without loss of generality that the vertex corresponding to the forget operation in $\mathrm{f}t'$ appears in G: $G \simeq \mathrm{f}G'$ for some graph G' such that $\mathcal{G}(t') \simeq G' \parallel \mathrm{l}H$.

  By induction hypothesis, we obtain parsings u' of G' and v' of $\mathrm{l}H$ such that $t' \equiv_k u' \parallel v'$. By the item 4. above, we get a parsing v of H such that $v' \equiv_k \mathrm{l}v$.

  We take $u = \mathrm{f}u'$. Using A13 we get $t \equiv_k \mathrm{f}(u' \parallel \mathrm{l}v) \equiv_k u \parallel v$. □

Together with Propositions 4.2.2 and 4.2.3, Proposition 4.3.1 makes it possible to normalise terms and to focus on the full prime ones.

An item is patently missing in the statement of Proposition 4.3.1: the one concerning 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 $k+1$; we need to restrict to those cases where the forgotten vertex is "safe" as defined next through the notion of forget point.

FORGET POINTS

4.4 Forget points on graphs.

Definition 4.4.1. Let G be a graph and x an inner vertex of G. We write (G,x) for the graph obtained from G by appending x to its interface; if (G,x) has treewidth at most k, we say that x is a *k-forget point of G*.

Lemma 4.4.2. *Every graph of treewidth and arity at most k in which there is an inner vertex contains at least a k-forget point.*

Proof. Let T be a tree decomposition of width at most k of a graph G containing an inner vertex.

Let b be a bag of T containing all sources of G. Since there exists an inner vertex in G, there exists in T a bag containing an inner vertex. Consider a shortest path P in T from b to a bag b' containing an inner vertex x of G. Let T' be the tree decomposition obtained by adding x to all bags of P. All bags of P but b' contain only sources. Since T has width at most k and G arity at most k, all bags of P but b' contain at most k vertices. Hence, bags of T' contain at most $k+1$ vertices and T' has width at most k.

The tree decomposition T' is also a tree decomposition of (G,x). So x is a k-forget point of G. □

Proposition 4.4.3. *Every non-atomic full prime graph of treewidth at most k contains at least a k-forget point.*

Proof. Let G be a non-atomic full prime graph of treewidth at most k. By Proposition 4.2.4, G has arity at most k. Since G is a non-atomic full prime graph, G contains an inner vertex. The statement follows by Lemma 4.4.2. □

4.5 Forget points on terms. Given a term $t \in \mathcal{T}_k(\mathcal{V})$ denoting a graph G with an inner vertex x, we say that t *reaches* x via $\equiv_k$ if there is a parsing t' of (G,x) such that $t \equiv_k \mathrm{f}t'$.

Since t' has width at most k, the vertex x reached by t is a k-forget point of G. Hence, all reached vertices are k-forget points. We typically drop the integer k in our notations, and simply state that a term reaches a forget point.

Proposition 4.5.1. *Every non-atomic full prime term reaches some forget point.*

Proof. Let t be a non-atomic full prime term of arity n. We prove by induction on t that there is a term u such that $t \equiv_k \mathrm{f}u$.

As t is non-atomic and full prime, t cannot be a variable, $\emptyset_n$, or a lift. In the other cases we proceed as follows:

- $t = \mathrm{f}u$; we are done.
- $t = t_1 \parallel t_2$; as G is prime, one of $\mathcal{G}(t_1)$ or of $\mathcal{G}(t_2)$ must be empty. Thanks to the A2 we can always assume that t_2 is empty. By Proposition 4.3.1, and A3, we get $t \equiv_k t_1 \parallel \emptyset_n \equiv_k t_1$. We conclude by induction hypothesis on t_1.

- $t = pv$; by induction hypothesis there is a term w such that $v \equiv_k \mathrm{f}w$. Using A9 we get $t \equiv_k p\mathrm{f}w \equiv_k \mathrm{f}\dot{p}w$ and we conclude with $u = \dot{p}w$. □

Corollary 4.5.2. *Every term of width and arity at most k containing a forget operation reaches a forget point.*

Proof. Let t be a term containing a forget operation. We prove there exists a term u such that $t \equiv_k \mathrm{f}u$. As t contains a forget operation, $\mathcal{G}(t)$ contains an inner vertex.

Let H be a non-atomic reduced component of $\mathcal{G}(t)$. It exists as $\mathcal{G}(t)$ contains an inner vertex. By Propositions 4.2.3 and 4.2.2, there exists a permutation p, a non-negative integer n, and a graph K such that

$$\mathcal{G}(t) \simeq p\mathrm{l}^n H \parallel K.$$

By Proposition 4.3.1, there are parsings v and w of H and K, respectively, such that

$$t \equiv_k p\mathrm{l}^n v \parallel w.$$

By Proposition 4.5.1, there exists a term v' such that $v \equiv_k \mathrm{f}v'$. Using A11 n times, and then A9 and A13, we get:

$$t \equiv_k \mathrm{f}(\dot{p}(r\mathrm{l})^n v' \parallel \mathrm{l}w)$$

and we conclude with $u = \dot{p}(r\mathrm{l})^n v' \parallel \mathrm{l}w$. □

SERIES DECOMPOSITIONS

4.6 Series decompositions on graphs. We introduce the *series operation*, slightly generalised compared to standard definitions, 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 swapping i and $k+1$.

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

We illustrate its behaviour at small arities in Figure 4.3.

We recover the standard series operation $\circ$ (see [1]) when the last argument is empty, and using a forget operation:

$$\circ(G_1, \dots, G_k) = \mathrm{fs}(G_1, \dots, G_k; \emptyset_{k+1})$$

We use this operation to decompose full prime graphs along a given inner vertex:

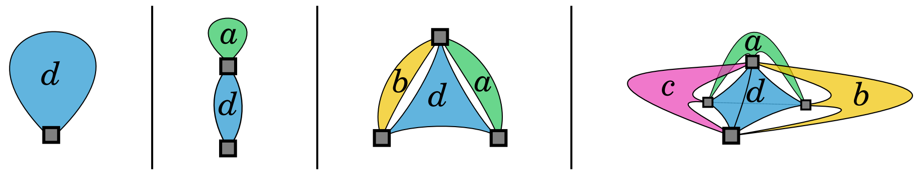


Figure 4.3: The series operations $s(;d)$, $s(a;d)$, $s(a,b;d)$, and $s(a,b,c;d)$.

Proposition 4.6.1 (Series decomposition)**.** *For every full prime graph G of arity k with an inner vertex x, there are unique graphs $G_1, \ldots, G_k, H$ such that $(G,x) \simeq s(G_1,\ldots,G_k;H)$ and*

1. *all prime components of H are full, and*
2. *if the jth source of a component of G_i is isolated, then $i < j$.*

In the *series decomposition* of G at x in the latter proposition, the graph G_i is called the *ith series argument* ($i = 1,\ldots,k$), and H the *series remainder*.

Proof. Decompose (G,x) into prime components, and classify these according to their isolated sources. Since G is full prime, x is never isolated in the prime components of (G,x).

Full prime components go into the series remainder.

Prime components in which the first source of G is isolated are put into the first series argument (under the permutation p_1, the first source gets swapped with x). Prime components in which the second source is isolated but not the first one go into the second series argument, etc. □

4.7 Series decompositions on terms.

Proposition 4.7.1. *If a term t reaches the last source of a graph of the form $s(G_1,\ldots,G_k;H)$, then there exist parsings $u_1,\ldots,u_k$, v of $G_1,\ldots,G_k$, H such that $t \equiv_k \mathrm{fs}(u_1,\ldots,u_k;v)$.*

Proof. Unfold the definition of series operation and conclude with Proposition 4.3.1. □

(IN)COMPLETENESS OF E_k

4.8 Completeness for easy graphs. The main challenge, in the rest of this chapter and in the next one, is to relate the different forget points of a given graph using axioms. We prove that the axiom set E_k is complete when there are few forget points.

A graph is called *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 lexicographic ordering $(|G|, |G| - \mathrm{ar}(G))$ of G's order with its number of inner vertices.

In the next proposition, we prove the completeness of E_k for easy graphs.

This first example provides a simple illustration of our method to prove completeness. Indeed, all our proofs of completeness share the same structure.

Proposition 4.8.1. *For all parsings t, u of an easy graph, we have $t \equiv_k u$.*

Proof. Let G be an easy graph t and u parsings of G. The proof proceeds by induction on $(|G|, |G| - \mathrm{ar}(G))$.

We distinguish several cases depending on whether G is non-atomic full prime or not:

- G is atomic and its sole edge is labelled a; by Proposition 4.3.1 we have $t \equiv_k a \equiv_k u$.
- G contains isolated sources; for some positive integer n, permutation p and full graph H with $|H| < |G|$ we have $G \simeq p\mathrm{l}^n H$. By Proposition 4.3.1, there exists parsings t' and u' of H such that $t \equiv_k p\mathrm{l}^n t'$ and $u \equiv_k p\mathrm{l}^n u'$. By induction hypothesis, $t' \equiv_k u'$.
- G is not prime; let H be a prime component of G. For some non-empty graph K we have $G \simeq H \parallel K$. By Proposition 4.3.1, there exists parsings t_H, u_H of H, and t_K, u_K of K such that $t \equiv_k t_H \parallel t_K$ and $u \equiv_k u_H \parallel u_K$. By induction hypothesis, $t_H \equiv_k u_H$ and $t_K \equiv_k u_K$.
- G is a non-atomic full prime graph; by Proposition 4.5.1 there are terms t' and u' such that $t \equiv_k \mathrm{f}t'$ and $u \equiv_k \mathrm{f}u'$. Since G is easy, t' and u' are parsings of (G, x) for x the only forget point of G. By induction hypothesis, $t' \equiv_k u'$. □

In this proof, the only part which is specific to easy graphs is the last item, where we use that easy graphs contain precisely one forget point per non-atomic reduced component. For non-easy graphs, we have to handle the cases where several forget points exist.

4.9 Completeness of E_0. Simple graphs of treewidth at most zero are precisely discrete graphs. Hypergraphs of treewidth at most zero are discrete graphs in which edges of arity zero and one are allowed. Hence:

Lemma 4.9.1. *A graph G of treewidth at most zero satisfies one the following statement:*

1. *G is empty,*
2. *G is atomic,*
3. *G is not prime,*
4. *G is the forget of an empty graph,*
5. *G is the forget of an atomic graph,*
6. *G is prime and the forget of non-prime graph.*

Proposition 4.9.2. *Given two terms $t, u \in \mathcal{T}_0(\mathcal{V})$,*

$$t \equiv_0 u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u).$$

Proof. By Proposition 3.14.1, it suffices to establish completeness. Let t and u be two parsings of width at most zero of a graph G.

The proof proceeds by induction on $\#G$. We distinguish three cases depending on which category of Lemma 4.9.1 G falls in:

1. G is empty, atomic, the forget of an atomic or empty graph, or prime and the forget of a non-prime graph; in any case G is easy and we conclude by Proposition 4.8.1.
2. G is not prime; let H be a prime component of G and K the non-empty graph such that $G \simeq H \parallel K$. By Proposition 4.3.1, there exists parsings t_H and u_H of H and t_K and u_K of K such that $t \equiv_0 t_H \parallel t_K$ and $u \equiv_0 u_H \parallel u_K$.

   By induction hypothesis, we have $t_H \equiv_0 u_H$ and $t_K \equiv_0 u_K$. □

4.10 Incompleteness of E_k when $k \geq 1$. Let a be a variable of arity $k+1$. We write AFG_k for the graph of arity $\mathcal{G}(\text{ff}a)$.

Depictions of AFG_k for several values of k are shown in Figure 4.4. The notation AFG stands for “Atomic Forget Graph”. This graph is the graph appearing in FA.

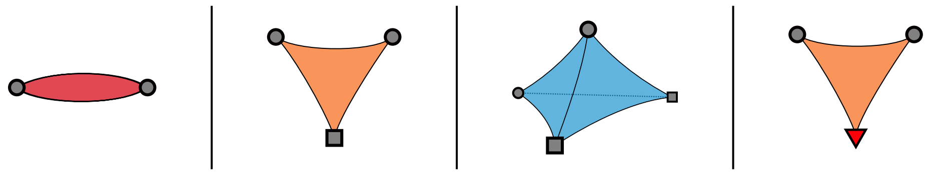


Figure 4.4: The graphs AFG_0, AFG_1, AFG_2, and AFG_k. For the latter, the red triangular source is a representation of all sources.

Proposition 4.10.1. *If $k \geq 1$ then there are terms t and u of width at most k with*

$$\mathcal{G}(t) \simeq \mathcal{G}(u) \quad \textit{and} \quad t \not\equiv_k u.$$

Proof sketch. We annotate forget operations by their corresponding vertices in terms.

Consider the parsings $t = \mathrm{f}_x\mathrm{f}_y a$ and $u = \mathrm{f}_y\mathrm{f}_x ra$ of AFG_k, with x and y its inner vertices and r the permutation swapping the last two sources.

The forget operations appearing in a term are in bijection with the inner vertices of the corresponding graph. Furthermore, the axioms A1-12 in E_k preserve the structure of forgotten vertices, and in particular, the order in which they are forgotten. Hence, only A13 can be used to reverse the order of f_x and f_y. Since AFG_k contains only one edge, which is incident to all vertices of the graph, when applying A13 to a parsing of AFG_k, the subgraph of AFG_k corresponding to the variable b in the axiom is necessarily empty. Hence applying A13 does not change the structure of forget operations in parsings of AFG_k and $t \equiv_k u$ does not hold. □

4.11 A sufficient condition for completeness. Recall from §3.10 that $\cong_k$ is the relation on terms of width at most k such that $t \cong_k u$ is $\mathcal{G}(t) \simeq \mathcal{G}(u)$.

Proposition 4.5.1 does not suffice to prove completeness as it gives no control on the reached forget points. Instead we would like:

Conjecture 4.11.1 (Reaching forget points)**.** *There exists a finite set $\mathrm{F}_k \subset \cong_k$ of equations such that full prime terms of width at most k reach all their forget points via $\equiv_{\mathrm{F}_k}$.*

The notation F stands for "Final Axioms".

We have already proved this conjecture true when $k = 0$ and we establish this for $k \leq 3$ across this and the following chapter. We do not know how to prove it directly when $k = 3$, however: we only get the conjecture *a posteriori*, from the completeness.

Instead, the cornerstone of our proof when $k = 3$ is the following: any two parsings of a non-atomic full prime graph may reach a common forget point:

Conjecture 4.11.2 (Forget point agreement)**.** *There exists a finite set* $\mathrm{F}_k \subset \cong_k$ *of equations such that two parsings* t, u *of width at most* k *of a non-atomic full prime graph reach a common forget point via* $\equiv_{\mathrm{F}_k}$.

Theorem 4.11.3. *The following are equivalent:*

1. *the treewidth axiomatisation conjecture,*
2. *Conjecture 4.11.1,*
3. *Conjecture 4.11.2.*

Proof. The implications 1. $\Rightarrow$ 2. and 2. $\Rightarrow$ 3. are direct.

We prove 3. $\Rightarrow$ 2.

Let $F = \mathrm{F}_k \cup \mathrm{E}_k$ where F_k is given by Conjecture 4.11.2.

By Proposition 3.14.1, and since $\mathrm{F}_k \subseteq \cong_k$, we need only prove completeness.

Let t and u be parsings of width at most k of a graph G. We prove $t \equiv_F u$ by induction on $(|G|, |G| - \mathrm{ar}(G))$ the lexicographic product of G's order with the number of inner vertices of G.

If the graph G has isolated sources, we use Proposition 4.2.2 and Proposition 4.3.1(3,4) to rewrite the parsings t and u into permuted lifts and proceed recursively.

Otherwise G is full, and we decompose it into prime components via Proposition 4.2.3:

- if G has no prime components then $G \simeq \emptyset_{\mathrm{ar}(G)}$ and according to Proposition 4.3.1(1) we have $t \equiv_F \emptyset_{\mathrm{ar}(G)} \equiv_F u$;
- if there exists at least two prime components in G then we use Proposition 4.3.1(5) on both parsings and we proceed recursively;
- otherwise G is full prime. Either it is atomic and we conclude by Proposition 4.3.1(2,3), or by Proposition 4.2.4, G has arity at most k. Hence t and u reach a common forget point x, and there exists parsings t' and u' of (G, x) such that $t \equiv_F \mathrm{f}t'$ and $u \equiv_F \mathrm{f}u'$. We proceed recursively on t' and u'. □

Hence, our conjectures are as strong as the problem asked by Courcelle and Engelfriet [12, p. 119] (of which the treewidth axiomatisation conjecture is a variation for a different graph syntax). They show what we believe to be an interesting path towards a positive answer to the treewidth axiomatisation conjecture. Indeed, our solution for the case of treewidth at most three consists in a proof of the corresponding instance of Conjecture 4.11.2.

CLIQUE POINTS

We introduce a class of vertices for which Conjecture 4.11.1 holds. In the next section, we use this result to answer the treewidth axiomatisation conjecture positively when $k = 1$.

Henceforth, $\equiv_k$ denotes the equational theory on terms of $\mathcal{T}_k(\mathcal{V})$ induced by the axiom set C_k. In other words, compared to previous sections, the axiom FA is allowed.

4.12 Clique points on graphs. An inner vertex x in a graph G is called a *clique point* if it is adjacent to every source of G.

Proposition 4.12.1. *Non-atomic full prime graphs of arity zero and one contain clique points.*

Proof. A non-atomic full prime graph of arity zero contains at least one inner vertex and the latter is a clique point.

In a non-atomic full prime graph of arity one the source has at least one neighbour, and the latter is a clique point. □

4.13 Clique points on terms. To prove that full prime terms reach their clique points, we first prove two lemmas. The first one intuitively makes it possible to zoom on a specific inner vertex by going under a forget operation. The second lemma is used to reach a vertex x on a term, by swapping forget operations, targeting the part of the corresponding graph x is in.

Lemma 4.13.1. *Let t be a term of arity at most $k-1$ with an inner vertex x in a full prime component. There is a term u such that $t \equiv_k \mathrm{f}u$ and x is either the last source of u or an inner vertex of a full prime component of u.*

Proof. We proceed by induction on the size of the component C of t containing x.

First, by Proposition 4.3.1 there exists a parsing t_x of C and a term t' such that $t \equiv_k t_x \parallel t'$. By hypothesis, t_x is non-atomic full prime. Hence Proposition 4.5.1 applies to get a term u such that $t_x \equiv_k \mathrm{f}u$.

If u is a parsing of $(\mathcal{G}(t_x), x)$, or if x is in a full prime component of u then we conclude using A13.

Otherwise x is in a component D of u that is smaller in size than C, but not full prime. We use Proposition 4.3.1 to find terms u_x and u' and a permutation p such that $u \equiv_k p\mathrm{l}^i u_x \parallel u'$, where u_x is a parsing of the reduced component associated to D and i is the number of isolated sources of D.

At this point we have:

$$t \equiv_k \mathrm{f}(p\mathrm{l}^i u_x \parallel u') \parallel t'.$$

As u_x is full prime and contains x we can use the induction hypothesis: there exists a term v such that $u_x \equiv_k \mathrm{f}v$ and either v's last source is x or x is in a full prime component of v. Let y be the last source of v; in subsequent term equations we write f_y for the forget operation that forgets y.

At this point we have:

$$\begin{aligned} t &\equiv_k \mathrm{f}(p\mathrm{l}^i \mathrm{f}_y v \parallel u') \parallel t' \\ &\equiv_k \mathrm{f}(p\mathrm{f}_y (r\mathrm{l})^i v \parallel u') \parallel t' && \text{(by A11)} \\ &\equiv_k \mathrm{f}(\mathrm{f}_y \dot{p}(r\mathrm{l})^i v \parallel u') \parallel t' && \text{(by A9)} \\ &\equiv_k \mathrm{f}\mathrm{f}_y((\dot{p}(r\mathrm{l})^i v \parallel \mathrm{l}u') \parallel \mathrm{l}t') && \text{(by A13)} \\ &\equiv_k \mathrm{f}_y\ \mathrm{f}r((\dot{p}(r\mathrm{l})^i v \parallel \mathrm{l}u') \parallel \mathrm{l}t'). && \text{(by FA)} \end{aligned}$$

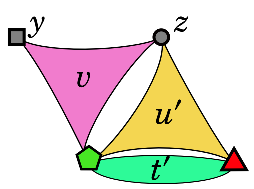
where $\dot{p}$ is the extension of p fixing the last element and r is the permutation swapping the last two elements (at various arities, here). Let w be the term

$$w = \mathrm{f}r((\dot{p}(r\mathrm{l})^i v \parallel \mathrm{l}u') \parallel \mathrm{l}t').$$

If x is v's last source, then $x = y$ and the derivation above proves $t \equiv_k \mathrm{f}w$ and w's last source is x.

Otherwise x is in a full prime component of v and to end the proof we show that it is also in a full prime component of w, i.e. that x is inner connected to all sources in w.

Call z the source forgotten by the forget operation appearing in the definition of w, and consider the shape of $(\mathcal{G}(t), y)$ given by the definition of w:



In this picture, the red triangle is used to represent all sources of t that are not sources of v (those corresponding to the operation $(r\mathrm{l})^i$), and the green pentagon is used to represent the common sources of t and v.

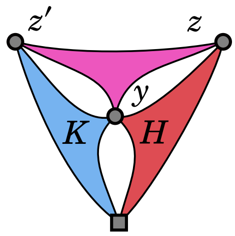


Figure 4.5: Illustration of Lemma 4.13.2 when $k = 3$ and $i = 2$. Let G be the depicted graph. If, in such a situation, x is ensured to be in either H or K, then we can swap z' or z with y, respectively. Assume that x is in H. The prime component of (G, z, y) containing x is smaller than that of (G, z, z'). This process can be applied to terms reaching z, z', and y, using FA. By induction, we reach x.

As x is in a full prime component of v, we easily get that x is inner connected to y, z, and to all sources represented by the green pentagon.

Furthermore, as x is in a full prime component of t, so is z, proving that z is connected in t by inner paths to every source represented by the red triangle. Observe that these paths remains inner in w. The concatenation in w of an inner paths from x to z with these paths provides inner paths from x to every source represented by the red triangle. Hence x is in a full prime component of w. □

An illustration of the next lemma is shown in Figure 4.5.

Lemma 4.13.2. *Let t be a parsing of a full prime graph G of arity $k + 1 - i$ ($i \geq 0$), and x an inner vertex of G.*

Assume the following holds: for every term u such that $t \equiv_k \mathrm{f}^i u$, if x is an inner vertex of a full prime component of $\mathrm{f}u$, then there exists a source s of u which is forgotten by one of the i forget operation in $\mathrm{f}^i u$ and x is in a component of u which is not incident to s.

Then, the term t reaches x.

Proof. Suppose we have a full prime term t of arity $k + 1 - i$ ($i \geq 0$), and an inner vertex x we want t to reach.

Thanks to Proposition 4.13.1 we know there exists a term v of arity k such that $t \equiv_k \mathrm{f}^i v$ and either x is a source of v, or x is in a full prime component A of v.

In the first case we conclude and in the latter case the proof proceeds by induction on the number of inner vertices of A (independently from i).

According to Propositions 4.3.1 and 4.5.1, v reaches some forget point y: there exists a parsing w of $(\mathcal{G}(v), y)$ such that $v \equiv_k \mathrm{f}w$. If $x = y$ then, by FA, t reaches x, and we conclude.

Otherwise, by hypothesis, in $\mathcal{G}(w)$, x is an inner vertex in a component C not incident to a source s of u which is not a source of G. By FA, there exists a term w' such that $t \equiv_k \mathrm{f}^j w'$ and C is a full prime component of w'. Since C is not incident to s, $j < i$. Also, every inner vertex of C is an inner vertex of A, but y is an inner vertex of A which not an inner vertex of C. Hence, we conclude by induction hypothesis. □

Proposition 4.13.3. *Full prime terms of width at most k reach their clique points.*

Proof. We prove that the hypothesis in Lemma 4.13.2 holds for clique points.

Let t be a full prime term and x a clique point of t, such that $t \equiv_k \mathrm{f}^i u$ for a term u of arity $k+1$, x is an inner vertex of a full prime component C of $\mathrm{f}u$. Let y be the last source of u.

By Proposition 4.2.4, the series remainder of u at y is a parallel composition of atomic graphs. Hence x is in a prime component C is not incident to all sources of u. Since x is a clique point, C is incident to all sources of u which are sources of t. Hence C is not incident to a source s of u which is not a source of t, that is, s is forgottent by one of the first i forget operation of the term $\mathrm{f}^i u$. □

(IN)COMPLETENESS OF C_k

4.14 Completeness of C_1. We answer the treewidth axiomatisation conjecture positively for terms of width at most one. The following solution uses all fifteen axiom schemes of C_1 (which induce thirty-seven axioms), and should be compared with the solution in §2.15 for which only seven axioms were used.

Theorem 4.14.1. *Given two terms $t, u \in \mathcal{T}_1(\mathcal{V})$,*

$$t \equiv_1 u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u).$$

Proof. This is a direct consequence of Theorem 4.11.3 combined with Propositions 4.2.4, 4.12.1 and 4.13.3. □

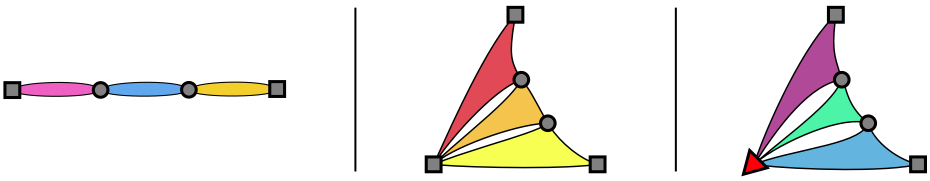


Figure 4.6: The graphs KG_2, KG_3, and KG_k; for the latter, the red triangular source is a representation of the last $k-2$ sources.

4.15 Incompleteness of C_k when $k \geq 2$. Assume $k \geq 2$. We call *kite operation* the graph operation mapping two graphs G and H both of arity k to the graph

$$G \bullet H = \circ(G, H, \emptyset_k, \dots, \emptyset_k).$$

We write KG_k ("Kite Graph of arity k") for the graph of arity k defined by two kite operations:

$$\mathrm{KG}_k = (a \bullet b) \bullet c$$

where a, b, c are atomic graphs of arity k.

Instances of KG_k for several values of k are depicted in Figure 4.6.

Proposition 4.15.1. *If $k \geq 2$ then there are terms t and u of width at most k with*

$$\mathcal{G}(t) \simeq \mathcal{G}(u) \quad \textit{and} \quad t \not\equiv_k u.$$

Proof sketch. In terms, we annotate the forget operations with their corresponding forget points. Let a, b, c be three edges of arity k.

Take $t = (a \bullet b) \bullet c$ and $t' = a \bullet (b \bullet c)$ two parsings of KG_k. Denote the two inner vertices of KG_k by x and y.

Unfolding the definitions gives:

$$t = \mathrm{f}_x\mathrm{s}(\mathrm{f}_y\mathrm{s}(x, y, \emptyset_k, \dots, \emptyset_k; \emptyset_{k+1}), z, \emptyset_k, \dots, \emptyset_k; \emptyset_{k+1}),$$

$$t' = \mathrm{f}_y\mathrm{s}(z, tf_x\mathrm{s}(y, z, \emptyset_k, \dots, \emptyset_k; \emptyset_{k+1}), \emptyset_k, \dots, \emptyset_k; \emptyset_{k+1}).$$

In t, the forget operation for x appears before the one for y, and in t' the forget operation for y appears before the one for x.

If $t \equiv_k u$ then, at some point in any corresponding proof, an axiom has to be applied to reverse the forget operations order. The only axioms available in C_k to do so are A13 and FA.

Let t'' be the parsing of KG_k on which one of these two axioms is applied.

If the axiom is FA, then using Proposition 4.3.1 one can prove $t'' \equiv_k \mathrm{f}_x\mathrm{f}_y t'''$. The term t''' is a parsing of width at most k of (KG_k, x, y), a graph of arity $k+2$, a contradiction.

If the axiom A13 is applied, that is, the following is applied up to a context and a substitution:

$$\mathrm{f}d \parallel e = \mathrm{f}(d \parallel \mathrm{l}e),$$

then the other forget operation must appear in e (otherwise the order of appearance of f_x and f_y is not changed by the axiom). The edge b of KG_k is incident to both x and y. In any corresponding parsing, x and y are simultaneously sources when b is added to KG_k. This is not the case of $\mathrm{f}d \parallel e$ if the second forget operation is in e, a contradiction. □

CHECKPOINTS

The equation $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ is a second example of forget axiom; it is denoted by FK, and the set $\mathrm{C}_k \cup \{\mathrm{FK}\}$ is denoted by A_k.

Henceforth, $\equiv_k$ denotes the equational theory induced by A_k on $\mathcal{T}_k(\mathcal{V})$.

We introduce a class of vertices for which Conjecture 4.11.1 holds. This results serves as a tool to establish Conjecture 4.11.1 for a broader class of vertices, and in the section after that to answer the treewidth axiomatisation conjecture positively when $k = 2$.

4.16 Checkpoints on graphs. 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 contains x. A *checkpoint* of G is a checkpoint between two sources of G.

This definition coincides with that from [10] at arity two.

The inner vertex of the kite operation of two atomic graphs is a checkpoint between the first and second sources. Moreover, a graph with a checkpoint has precisely a kite shape:

Proposition 4.16.1. *Let G be a graph with a and b two variables of arity $\mathrm{ar}(G)$. An inner vertex x of G is a checkpoint of G if and only if there exists a permutation p such that G has shape $p(a \bullet b)$ and the inner vertex of this shape is x.*

Proof. We prove the difficult implication which is the direct implication. Let x be a checkpoint in a graph G.

Up to replacing G with pG for some permutation p, we can assume that x is a checkpoint between the first and second sources of G.

Fix an integer $i \geq 3$. If the ith series argument of G at x is non-empty, then it induces an inner path from the first to the second source of G free of x. Combining the definition of the kite operation with Proposition 4.6.1 applied to (G, x) proves that G has shape $p(a \bullet b)$. □

Proposition 4.16.2. *Let G be a full prime graph with two inner vertices x and y. If x is in the ith series argument at y, then y is a checkpoint between x and the ith source in G.*

Proof. By definition, the ith series argument of a series decomposition of G does not contain its ith source. Let P be an inner path from x to the ith source of G. It must start in the ith series argument at x and end outside at the ith source, as was just observed. As P is inner, it can only get out from the ith series argument at y. □

Proposition 4.16.3. *Let G be a graph with two inner vertices x and y and two designated sources s_x and s_y. If x is a checkpoint between y and s_y and y a checkpoint between x and s_x, then x and y are both checkpoints between s_x and s_y.*

Proof. By symmetry, it suffices to show that x is a checkpoint between s_x and s_y. Let P be an inner path from s_x to s_y. Consider a shortest y-P inner path Q in G. Such a path exists as G is full prime and y is thus inner connected to any inner element of P. Say Q is from y to z for some z in P. The inner path QzP goes from y to s_y, so that it must contain x by assumption. If x belongs to zP then this completes the proof. Otherwise x belongs to Qz. In that case, the inner path $xQz\overline{P}$ goes from x to s_x and must contain y by assumption, which contradicts the minimality assumption about Q. □

Proposition 4.16.4. *Let G be a full prime graph of arity and treewidth k. If G contains a checkpoint x as well as a forget point $y \neq x$, then y and x are both checkpoint of G for the same pair of sources.*

Proof. Consider the series decompositions of G at y and x and apply Proposition 4.16.2.

At x, by Proposition 4.16.1, only two series arguments are non-empty and x is a checkpoint between y and a source s_x of G.

At y, by Proposition 4.2.4, x cannot be in the series remainder. Hence y is a checkpoint between x and a source s_y of G.

Conclude by Proposition 4.16.3. □

4.17 Checkpoints on terms.

Proposition 4.17.1. *Full prime terms reach all their checkpoints.*

Proof. Let t be a full prime term of arity $k - i$ ($i \geq 0$) containing a checkpoint x.

We proceed by induction on the lexicographic product $(|G|, |G| - k)$ of G's order with the number of inner vertices in G.

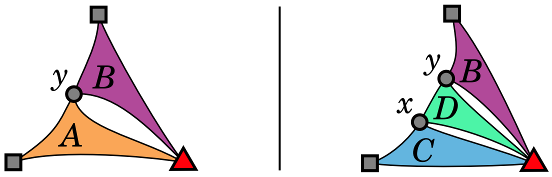
By Lemma 4.13.1, there exists a term u of arity k such that $t \equiv_k \mathrm{f}^i u$ and x is either a source of u, in which case we conclude with A13, or in a full prime component C of u.

In the latter case, by Proposition 4.3.1, there exists a parsing u_C of C and a term u' such that $u \equiv_k u_C \parallel u'$. Observe that x is a checkpoint of u_C.

If u' is non-empty or $i > 0$ then we proceed by induction on u_C and with A13. Otherwise, t has arity k. By Proposition 4.5.1, t reaches some forget point y. If $x = y$ we are done.

Otherwise, by Proposition 4.16.4, x and y are checkpoints between the same sources of t.

By Proposition 4.3.1, it is possible to assume x and y are checkpoints between the first and second sources. Hence G has the following two shapes:



where A, B, C, and D denote the subgraphs of G which substitute the corresponding labelled edges in the shapes, and where the red triangular source represents the last $k - 2$ sources of the graph.

Since t reaches y we have a parsing t' of (G, y) such that $t \equiv_k \mathrm{f}t'$.

According to Proposition 4.3.1 and 4.7.1, there exist parsings t_A and t_B of A and B such that

$$t \equiv_k t_A \bullet t_B.$$

Observe that x is a checkpoint between the first and second sources in A. By induction hypothesis, t_A reaches x. Applying Proposition 4.3.1 and 4.7.1 again, there exists parsings t_C and t_D of C and D, respectively, such that

$$t_A \equiv_k t_C \bullet t_D.$$

Using FK, we get:

$$t \equiv_k t_C \bullet (t_D \bullet t_B).$$

Unfolding the definition of the first operation $\bullet$,

$$t \equiv_k \mathrm{fs}(t_C, t_D \bullet t_B, \emptyset_k, \dots, \emptyset_k; \emptyset_{k+1})$$

proving that t reaches x. □

ANCHORS

We introduce a class of vertices which generalises cutvertices in the case of sourced graphs, and for which Conjecture 4.11.1 holds. Thanks to this result we are able to answer the treewidth axiomatisation conjecture positively when $k = 2$, and to reduce the case where $k = 3$ to anchor-free graphs. The next chapter is devoted to the study of anchor-free graphs of treewidth at most three, and as a consequence we answer the treewidth axiomatisation conjecture positively when $k = 3$.

4.18 Anchors on graphs.

Definition 4.18.1. An *anchor* in a graph G is an inner vertex x such that the series remainder of G at x is either

1. empty, or
2. contains at least two full prime components.

An anchor satisfying in the ith condition of the definition is called an anchor of the *ith kind*.

To aid recall, observe that anchors of kind "2" are associated with "2" full prime components.

Checkpoints are anchors of the first kind by Proposition 4.16.1.

Cutvertices, as defined in §1.18, are anchors of the second kind at arity zero.

Anchors are illustrated in Figure 4.7.

We prove that all forget points are either clique points or anchors at arity k. The converse holds at all arities for full prime graphs, but we only get it *a posteriori*, as a consequence of Proposition 4.19.1, which proves that full prime terms reach their anchors.

Proposition 4.18.2. *At arity k, all k-forget points are either clique points or anchors of the first kind.*

Proof. If x is a forget point of a graph G of arity k, the full prime components of (G, x) must be atomic by Proposition 4.2.4. Thus, if x is not a clique point, then it is an anchor of the first kind. □

Corollary 4.18.3. *Every non-atomic full prime graph of arity k contains a clique point or an anchor.*

Proof. By Propositions 4.18.2 and 4.4.3. □

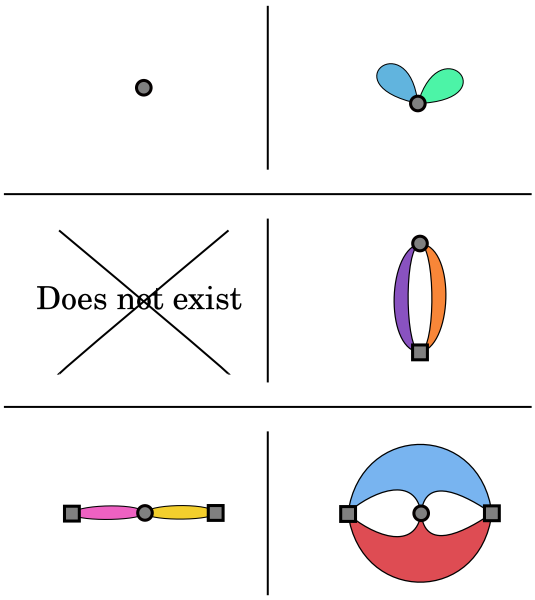


Figure 4.7: Anchors of the first and second kinds (on the left and right-hand sides, respectively) at arity zero, one, and two.

Lemma 4.18.4. *In a full prime graph of arity k and treewidth at most k, the series remainder at each anchor cannot contain inner vertices.*

Proof. Let G be a full prime graph of arity k and treewidth at most k with an anchor x.

For the sake of contradiction, assume that the series remainder at x contains an inner vertex y. In particular x cannot be an anchor of the first kind: x is an anchor of the second kind, and the series remainder of G at x contains at least two full prime components.

Write S for the set of sources of G.

We work with the skeleton H of G and prove that H contains K_{k+2} as a minor, a contradiction with G having treewidth at most k.

Denote by C the prime component of the series remainder at x containing y, and by D another prime component of the series remainder.

In $H - S - x$, the vertices of C and D form components C_H and D_H which are, in H, both incident to all sources in S as well as to x.

In H, contract all edges of C_H, and of D_H, and remove all vertices not in C_H, D_H, S, or distinct from x. In the resulting graph are the sources in S, x, as well as two vertices x_C and x_D witnessing the contractions of C_H and D_H, respectively. Finally, contract the edge connecting x with x_D (this edge exists as D being a prime component of the series remainder, it is incident to x in G).

We claim that the minor M of H obtained by this procedure is K_{k+2}.

All edges between vertices of S in M exist by definition of skeleton. Furthermore, x_C is adjacent to all vertices in S as well as to x thanks to C being a prime component of the series remainder of G at x. For similar reasons, x_D above was adjacent to all vertices of S. Finally, x is adjacent to all vertices of S thanks to the contraction of xx_D. □

Proposition 4.18.5. *Let G be a full prime graph of arity k and treewidth at most k.*

If G has two distinct anchors x and y, then they are both checkpoints between the same pair of sources.

Proof. By Lemma 4.18.4, the anchor x (resp. y) cannot be in the series remainder of G at y (resp. x): it must be in a series argument of the series decomposition at y (resp. x), say the ith (resp. jth) one.

According to Proposition 4.16.2, x (resp. y) is a checkpoint between y (resp. x) and the ith (resp. jth) source of G.

We conclude with Proposition 4.16.3. □

This proposition states that a full prime graph of maximal arity with two anchors has a KG_k-shape.

4.19 Anchors on terms.

Proposition 4.19.1. *Full prime terms reach all their anchors.*

Proof. Let t be a full prime term of arity $k+1-i$ ($i \geq 0$) with an anchor x.

We prove that t reaches x via Lemma 4.13.2 and Proposition 4.17.1.

More precisely, we prove that if $t \equiv_k \mathrm{f}^i u$ and x is an inner vertex of a full prime component of $\mathrm{f}u$, then x is either a checkpoint in $\mathrm{f}u$ (in which case t reaches x via Proposition 4.17.1 and A13) or in a component of u which is not incident to a source of u that is forgotten by one of the i forget operations in $\mathrm{f}^i u$ (in which case we apply Lemma 4.13.2).

First, by Proposition 4.2.4, x is not in a full prime component of u. Assume that x is in a prime component D of u which is incident to all sources of u which are forgotten by the i forget operations in $\mathrm{f}^i u$. We prove that x is a checkpoint in $\mathrm{f}u$.

We reason by case analysis on the kind of x as an anchor of t. We write $x_1, \ldots, x_{i-1}$ for the sources of u forgotten by the operations f^{i-1} in $t \equiv_k \mathrm{f}^{i-1}\mathrm{f}u$. Write y for the source of u forgotten in $\mathrm{f}u$.

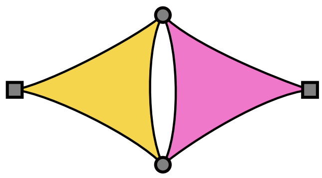
Since D is not full prime, there exists a source s of u which is not incident to D, and since s is not forgotten by the i forget operations in $\mathrm{f}^i u$, s is a source of t. We have that y is a checkpoint between x and s in $\mathrm{f}u$.

If x is an anchor of the first kind then since there are no full prime component in $\mathcal{G}(t) - x$, there exists a source s' of t such that x is a checkpoint between y and s' in t, and by extension in $\mathrm{f}u$. By Proposition 4.16.3, both x and y are checkpoints of $\mathrm{f}u$.

Otherwise, x is an anchor of the second kind in t: there are at least two full prime components C_1 and C_2 in $\mathcal{G}(t) - x$. Assume that y is in C_1 (and hence not in C_2). Being full prime, C_2 provides us with an inner path P in t from x to the source s of t that does not contain y. Since y is a checkpoint between x and s in $\mathrm{f}u$, P must contain a source s' of $\mathrm{f}u$ among $x_1, \dots, x_{i-1}$. Consider an inner path Q from y to s' in $\mathrm{f}u$; Q is also inner in t, starts in C_1, and ends in C_2: it must go through x. Hence x is a checkpoint between y and s' in $\mathrm{f}u$. By Proposition 4.16.3, both x and y are checkpoints of $\mathrm{f}u$. □

Proposition 4.19.1 implies Conjecture 4.11.1 full prime terms containing an anchor. Therefore, once Proposition 4.19.1 proved, it remains to prove Conjecture 4.11.2 for parsings of anchor-free graphs. This is the topic of the next chapter when $k = 3$.

There exists graphs without anchors, even at treewidth at most two:



Nonetheless, in this example, the graph has a clique point.

(IN)COMPLETENESS OF A_k

4.20 Completeness of A_2**.** We give a positive answer to the treewidth axiomatisation conjecture for terms of width at most two.

Theorem 4.20.1. *Given two terms $t, u \in \mathcal{T}_2(\mathcal{V})$,*

$$t \equiv_2 u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u).$$

Proof. This is a direct consequence of Theorem 4.11.3 combined with Propositions 4.2.4, 4.12.1, 4.13.3, 4.18.2, and 4.19.1. □

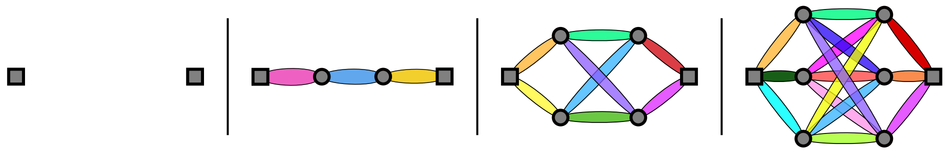


Figure 4.8: The graphs XG_1, XG_2, XG_3, and XG_4. Observe that $\mathrm{XG}_1 \simeq \emptyset_2$ and $\mathrm{XG}_2 \simeq \mathrm{KG}_2$.

A similar result is proved in [10, 14], for a different syntax that includes the series operation and limits the arity to two instead of three. In [10], they use the fact that graphs of treewidth at most two are precisely those excluding K_4 as a minor, and [14] uses a graph rewriting system recognising graphs of treewidth at most two.

Both characterisations exist at treewidth at most k, albeit the exact lists of minors and rewriting rules are unknown whenever $k \geq 4$ and $k \geq 5$, respectively.

Our proof uses neither characterisation of graphs of treewidth at most two. Moreover, in §5.10, we retrieve the structural characterisation via K_4 as a consequence of our positive answer to the treewidth axiomatisation conjecture for $k = 3$.

4.21 Incompleteness of A_k when $k \geq 3$. A *bipartite graph* is a graph G whose vertex set can be partitioned in two sets U and V such that if two vertices are adjacent then one lies in U and the other in V. The pair (U, V) is called the *bipartition* of G. A *complete bipartite graph* is a bipartite graph with bipartition (U, V) containing one edge incident to x and y for every vertices $x \in U$ and $y \in V$, and no other edges. We write $K_{n,m}$ for the complete bipartite graph with a bipartition of two sets of n and m vertices, respectively.

Assume $k \geq 1$. We write XG_k for the graph of arity two with $2(k-1)$ inner vertices such that:

- all edges of XG_k are binary edges,
- The subgraph induced by the inner vertices is the complete bipartite graph $K_{k-1,k-1}$. Denote the corresponding bipartition by (A, B),
- one source of XG_k is adjacent to the vertices of A, and the other to the vertices of B.

Examples of XG_k for several values of k are shown in Figure 4.8. The X in XG represents the bipartition of the inner vertices.

Proposition 4.21.1. *The graph* XG_k *has treewidth at most* k.

Proof. Let (A, B) be the bipartition of the inner vertices. In the graph (G, A) (so G in which all vertices of A have been promoted to sources), prime components are edges or stars S_3 with three branches. All such components have treewidth at most two and (G, A) has arity $k + 1$.

By substituting parsings of width at most two of its prime components in the full prime decomposition of (G, A), we obtain a parsing of (G, A) of width k. Adding $k - 1$ forget operations gives a parsing of width k of G. □

Proposition 4.21.2. *If* $k \geq 3$ *then there are terms* t *and* u *of width at most* k *with*

$$\mathcal{G}(t) \simeq \mathcal{G}(u) \quad \text{and} \quad t \not\equiv_k u.$$

Proof sketch. In terms, we index the forget operations by their corresponding forget points.

We take $G = \mathrm{XG}_k$ (with (A, B) the bipartition of the inner vertices) and t and u parsings starting with $k - 1$ forget operations, those of t corresponding to the vertices of A and those of u corresponding to the vertices of B (see the proof of Proposition 4.21.1 for more details on these parsings).

For the sake of contradiction, assume that $t \equiv_k u$. This implies the use of an axiom to reverse the order of appearance of the forget operations corresponding to A and B.

The set A_k contains three axioms suited for such a purpose: A13, FA, and FK.

If A13 is used, then the subterm concerned with this axiom is a parallel of a forget of a term with another term:

$$\mathrm{f}_x t' \parallel u' \equiv_k \mathrm{f}_x(t' \parallel \mathrm{l}u').$$

Since applying this axiom changes the order of appearance of forget operations corresponding respectively to a vertices x in A and y in B, we have that f_y appears in u'. Since x and y are adjacent in XG_k, the corresponding edge appears in every parsing of XG_k. Since it is incident to x, it appears in $\mathrm{f}_x t'$, and since it is incident to y, it appears in u'. Both statement cannot be simultaneously true, providing a contradiction.

If FA is used, it must, at some point, be used on the outer forget operation, so that up-to applying Proposition 4.3.1, having $t \equiv_k u$ implies a parsing of XG_k of shape $\mathrm{f}_x\mathrm{f}_y t'$ with $x \in A$ and $y \in B$ and t' a parsing of (G, x, y). By Lemma 4.13.1, there exists a term t'' such that $t' \equiv_k \mathrm{f}^{k-4} t''$. Because $x \in A$ and $y \in B$, there exists at least two vertices $x' \in A$ and $y' \in B$ which are inner in t''. By definition of XG_k, those are in a full prime component of t'', a contradiction with Proposition 4.2.4.

If FK is used, then in the subterm concerned with this axiom, the two vertices corresponding to the forget operations of FK have to be checkpoints between two sources. There are only two relevant cases in which a vertex $x \in A$ can be a checkpoint a subterm of a parsing of XG_k; between either

- the two sources of XG_k in $(G, A \setminus \{x\})$. No other checkpoint between the same sources exist in $(G, A \setminus \{x\})$, or,
- one source of XG_k and a vertex in B in $(G, A \setminus \{x\})$. Again no other such checkpoint can exist.

Since the axiom FK supposes two checkpoints, and each time only one exists, it cannot be used. □

HARD GRAPHS AND MINIMAL SEPARATION TUPLES

4.22 Hard graphs. We call *hard* the non-atomic full prime graphs which are clique point-free and anchor-free.

Hard graphs of treewidth at most k have their arity in $\{2, \ldots, k-1\}$ by Propositions 4.12.1 and 4.18.3. An example of a hard graph is XG_k.

By Propositions 4.3.1, 4.13.3, and 4.19.1, proving Conjecture 4.11.1 or 4.11.2 for hard graphs is sufficient to obtain a positive solution to the treewidth axiomatisation conjecture. We do so in the next chapter in the case of graphs of treewidth at most three.

To reduce the number of cases in proofs of the next chapter, we introduce a notion of minimal forget point. Considering hard graphs ensures that the corresponding preorder is a well-founded partial order.

4.23 Minimal forget points on graphs. Let x and y be inner vertices of a graph G. We write $y \preceq x$ when y is a checkpoint between x and some source. We write $\prec$ for the irreflexive restriction of $\preceq$. An inner vertex x of G is called *minimal* when it is so with respect to $\prec$, that is, there does not exist a vertex y in G such that $y \prec x$.

We write $S(x)$ for the series remainder of a graph at some inner vertex x.

Proposition 4.23.1. *For all hard graphs, $\preceq$ is a partial order, and $\prec$ is well-founded.*

Proof. Let G be a hard graph.

- **Reflexivity** of $\preceq$: every path from an inner vertex x to a source of G starts with x. Thus $x \preceq x$ holds;

- **Antisymmetry** of $\preceq$: for the sake of contradiction, suppose that x and y are two distinct vertices such that $x \preceq y$ and $y \preceq x$ but $x \neq y$.

  According to Proposition 4.16.3, x and y are checkpoints of G between the same pair of sources.

  Checkpoints being anchors, G is not hard, a contradiction.

- **Transitivity** of $\preceq$: assume $x \preceq y$ and $y \preceq z$ for x, y, and z three vertices. We prove $x \preceq z$.

  Let $1 \leq i, j \leq \mathrm{ar}(G)$ be the integers such that, in G, x is a checkpoint between y and the ith source and y is a checkpoint between z and the jth source.

  If x is a checkpoint between z and the ith source then we are done.

  Otherwise there is a path P from z to the ith source which does not contain x. As x is a checkpoint between y and the ith source, any path Q from y to z contains x as otherwise QP would induce a path from y to the ith source not containing x. Hence x is a checkpoint between y and z. As any path from z to the jth source contains y, it also contains x, and x is a checkpoint between y and the jth source.

- **Well-foundedness** of $\prec$: observe that if $y \prec x$ then $|S(y)| < |S(x)|$. Hence it is well-founded. □

4.24 Minimal forget points on terms.

Proposition 4.24.1. *If $y \prec x$ in a full prime graph G, then every parsing of G reaching x also reaches y.*

Proof. Direct consequence of Proposition 4.17.1. □

Corollary 4.24.2. *Every non-atomic full prime term reaches some minimal forget point.*

Proof. Direct consequence of Propositions 4.5.1, 4.23.1, and 4.24.1. □

4.25 Minimal separation tuples on graphs. Following the parallel between cutvertices in simple graphs (§1.18) and anchors in sourced graphs (§4.18), we aim for separators of size two in hard graphs. More precisely, we prove next that, at arity $k-1$, we are able to reach any vertex in a "separator" of size two, for an appropriate notion of separator.

Instead of focusing on separation pairs in hard graphs of treewidth at most three (which suffices to solve the treewidth axiomatisation conjecture at

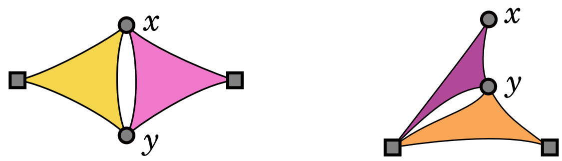


Figure 4.9: A separation pair ($y \diamond x$), and partial order on vertices ($y \prec x$).

treewidth at most three), we introduce the general notion of separation tuples in graphs of treewidth at most k, even though we are not able to conclude in this general setting.

Let G be a graph and $x_1, x_2, \dots, x_n$ be the inner vertices of G ($1 \leq n \leq |G| - \mathrm{ar}(G)$). We refer to $(x_1, x_2, \dots, x_n)$ as a *k-forget tuple* of G if $(G, x_1, x_2, \dots, x_n)$ has treewidth at most k.

Let t be a parsing of G. If there exists a parsing t' of $(G, x_1, x_2, \dots, x_n)$ such that $t \equiv_k \mathrm{f}^n t'$, we say that t *reaches* the tuple $(x_1, x_2, \dots, x_n)$.

By definition, a term may only reach forget tuples, and, according to FA, a term reaches $(x_1, x_2, \dots, x_n)$ if and only if it reaches $(x_{p(1)}, x_{p(2)}, \dots, x_{p(n)})$ for any permutation p of $[1, \dots, n]$; it follows that every term reaching the tuple $(x_1, x_2, \dots, x_n)$ reaches x_j for every $1 \leq j \leq n$.

The tuple $(x_1, x_2, \dots, x_n)$ is called a *separation tuple* if there are no full prime component in $G - x_1 - x_2 - \cdots - x_n$, In this case we write $\diamond(x_1, x_2, \dots, x_n)$. For separation pairs, we write $x \diamond y$.

Observe that separation singletons coincide with anchors of the first kind.

Each side of the bipartition of the inner vertices of XG_k is a k-forget separation tuple. Figure 4.9 shows a separation pair as well as two vertices x and y with $y \prec x$.

Proposition 4.25.1. *For all inner vertices $x_1, x_2, \dots, x_n, y$ in a hard graph G, if $y \prec x_n$ and $\diamond(x_1, \dots, x_n)$ then $\diamond(x_1, \dots, x_{n-1}, y)$.*

Proof. For the sake of contradiction, let C be a full prime component in $G - x_1 - \cdots - x_{n-1} - y$; C must contain x_n as there are no full prime components in $G - x_1 - \cdots - x_n$ and nor in $G - x_1 - \cdots - x_{n-1} - y - x_n$. Hence, for all sources s of G there exists a path from x_n to s in C (and in $G - x_1 - \cdots - x_{n-1} - y$ by extension), a contradiction with $y \prec x_n$. □

In a hard graph, a tuple of vertices is *minimal* when all its vertices are so with regard to $\prec$.

Each side of the bipartition of the inner vertices of XG_k is a minimal separation tuple.

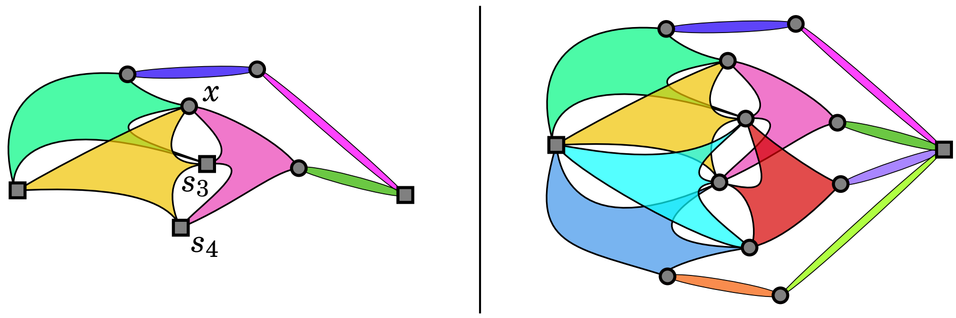


Figure 4.10: Let G be the graph of the left-hand side. Its vertex x is an anchor of the first kind; it is not difficult to check that G is easy at treewidth at most four. The graph H on the right-hand side is $\mathrm{ff}(G \parallel rG)$ where r is the permutation swapping three and four. In H, there are no clique points, no anchors, and no forget separation tuples (at treewidth at most four). Moreover, H is hard.

It is false that forget separation tuples always exists in hard graphs, as shown in Figure 4.10.

Next, we focus on separation pairs in graphs of arity $k-1$.

Proposition 4.25.2. *Let x be a forget point in a hard graph of treewidth at most k and arity $k-1$. For every forget points y of $S(x)$, either $y \prec x$ or $y \diamond x$ holds.*

Proof. We classify the prime components of (G,x,y) according to the sources they are incident to.

Full prime components cannot exist in (G,x,y) as, according to Proposition 4.2.4, they would be edges, and both x and y would be clique points in G.

Components of (G,x,y) incident to all sources but y cannot exist. Indeed, together with the series remainder $S(x)$ of G at x (containing y), they would form two full prime components in (G,x), proving that x is an anchor.

We distinguish two cases depending on the existence of some components of (G,x,y):

- there exists at least one prime component C in (G,x,y) incident to all sources but x; x cannot be inner connected to all sources in (G,y), as otherwise, its component together with C would provide two full prime components in (G,y), proving that y is an anchor. Hence y is a checkpoint between x and a source of G, and $y \prec x$.

- there are no full prime component in (G, x, y) incident to all sources but x; as G is full prime, any component of $G - x - y$ induces a prime component of (G, x, y) incident to either x or y. In particular, a full prime component of $G - x - y$ induces either a full prime component of (G, x, y), or a component incident to all sources but either x or y. Thhose being non-existent in (G, x, y), there cannot exist full prime component in $G - x - y$, and $y \diamond x$. □

Corollary 4.25.3. *Every hard graph of treewidth at most k and arity $k - 1$ contains a minimal separation pair.*

Proof. By Propositions 4.4.3, 4.24.1, and 4.23.1, there exists a minimal forget point x in G, and then a forget point y in (G, x), minimal in G. By Proposition 4.25.2, since both x and y are minimal, we have $x \diamond y$. Hence, (x, y) is a minimal separation pair of G. □

Lemma 4.25.4. *Let (x, y) be a separation pair in a graph G of arity two. All paths from one source to the other in G must visit either x or y.*

If furthermore G is hard then there exists in G a path from one source to the other that contains x (resp. y) but not y (resp. x).

Proof. By definition of separation pairs, and using the fact that hard graphs are full prime and do not have checkpoints. □

4.26 Minimal separation pairs on terms.

Lemma 4.26.1. *If $\diamond(x_1, x_2, \ldots, x_{i-1}, y)$ in a full prime graph G, then y is an anchor in $(G, x_1, \ldots, x_{i-1})$, and every parsing of G reaching $(x_1, x_2 \ldots, x_{i-1})$ also reaches $(x_1, \ldots, x_{i-1}, y)$.*

Proof. There exists no full prime component in $G - x_1 - \cdots - x_{i-1} - y$, and, by extension, in $(G, x_1, \ldots, x_{i-1}, y)$ neither. Thus, y is an anchor of the first kind in $(G, x_1, \ldots, x_{i-1})$.

The second part of the statement follows by Proposition 4.19.1. □

Proposition 4.26.2. *Every hard term of treewidth at most k and arity $k - 1$ reaches some minimal separation pair.*

Proof. Let t be a hard term of treewidth at most k and arity $k - 1$.

By Corollary 4.24.2, t reaches a minimal forget point x: there exists a parsing t' of (G, x) such that $t \equiv_k \mathrm{f}t'$. In t', by Proposition 4.3.1, we can isolate the series remainder of t at x. Using Corollary 4.5.2, the corresponding parsing of this series remainder reaches a forget point y. By A13, t reaches (x, y). Thanks to Lemma 4.26.1, we can assume y to be minimal in G.

Since both x and y are minimal, by Proposition 4.25.2 (x, y) is a minimal separation pair. □

5
An axiomatisation of graphs of treewidth at most three

We provide a positive answer to the treewidth axiomatisation conjecture for graphs of treewidth at most three. We rely on the succession of results proved in Chapter 4. This allows us to focus on hard terms of width at most three and arity two, which reach separation pairs by Proposition 4.26.2.

In the first section (§5.1 and 5.2) we extend A_3 and state the main result.

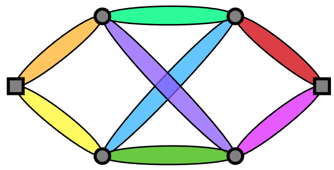
In a second section (§5.3 to §5.5), we show how to relate distinct separation pairs of a hard term of width at most three. More precisely, we show that there are a few shapes a hard term of width at most three with distinct separation pairs must have.

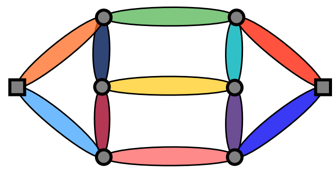
In the third section (§5.6 and 5.7), we study a result characterising graphs without K_3 as a minor, with an added constraint that the vertices of K_3 must correspond to the sources of the graph. This condition is formalised via *sourced minors*, a notion that is related to the standard notion of rooted minors—an important tool in the graph minor theory of Robertson and Seymour.

The characterisation of graphs without K_3 as a sourced minor is then used, in a fourth and final section (§5.8 to §5.9), to refine the shapes hard terms of width at most three with distinct separation pairs have. This refinement is then used to complete our answer to the treewidth axiomatisation conjecture when $k = 3$.

STATEMENT

5.1 The axioms set F_3. Consider the following two graphs XG_3 and DG of treewidth three (DG stands for "Domino graph"):





Observe that each of these two graphs contains four forget points which are the vertices of the only two (minimal forget) separation pairs.

Let G be a graph in $\{\mathrm{XG}_3, \mathrm{DG}\}$ with x and y two inner vertices in distinct separation pairs of G. Let $\mathrm{f}t_x = \mathrm{f}t_y$ be an equation where t_x (resp. t_y) is a parsing of (G, x) (resp. (G, y)). We write FX for the equation corresponding to XG_3, and FD for that corresponding to DG, and we denote $\mathrm{A}_3 \cup \{\mathrm{FX}, \mathrm{FD}\}$ by F_3.

Henceforth, $\equiv_3$ denotes the equational theory on $\mathcal{T}_3(\mathcal{V})$ induced by F_3.

The specific parsings of XG_3 and DG defining FX and FD do not matter, as a consequence of Proposition 4.8.1 as well as:

Proposition 5.1.1. *For every forget point x of a graph $G \in \{\mathrm{XG}_3, \mathrm{DG}\}$, (G, x) is easy.*

Proof. In the following arguments, in order to see that an inner vertex cannot be a forget point, it often suffices, according to Proposition 4.2.4, to show that it would give a full prime graph of arity four which is not atomic.

Let x be a forget point of G. Observe that (G, x) is full prime and its only forget point is the vertex x' such that (x, x') is a separation pair of G.

- For XG_3, (XG_3, x, x') has four reduced components: two edges and two graphs which are easy since they have only one inner vertex.
- For DG, (DG, x, x') has three reduced components: two edges, and the remainder of the graph. Note C denote the latter. There is a single forget point in C: the inner vertex y adjacent to all other inner vertices of C. In (C, y) there are three reduced components, each with only one inner vertex. □

5.2 Completeness of F_3. In this chapter we provide a positive answer to the treewidth axiomatisation conjecture when $k = 3$:

Theorem 5.2.1. *Given two terms $t, u \in \mathcal{T}_3(\mathcal{V})$,*

$$t \equiv_3 u \quad \Leftrightarrow \quad \mathcal{G}(t) \simeq \mathcal{G}(u).$$

By Theorem 4.11.3 and Proposition 4.26.2, the following proposition implies Theorem 5.2.1.

Proposition 5.2.2. *Let G be a hard graph of arity two and treewidth at most three. If a parsing of G reaches a minimal separation pair, then it reaches any other minimal forget separation pair.*

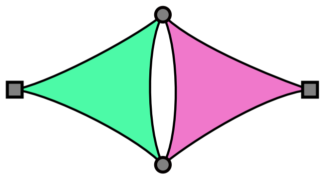
The rest of this chapter is dedicated to its proof.

COMBINING SEPARATION PAIRS

5.3 Combining shapes. Given a graph G having shapes H_1 and H_2 via substitutions σ_1 and σ_2, respectively, we say that a shape H of G (via a substitution σ) *combines* H_1 and H_2 when H has shapes H_1 and H_2 via substitutions σ_1' and σ_2', respectively, and:

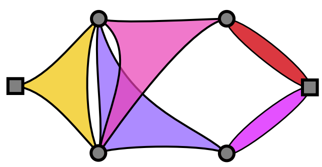
$$\sigma = \hat{\sigma}_1 \circ \sigma_1' = \hat{\sigma}_2 \circ \sigma_2'.$$

A graph G of arity two with a separation pair has the following shape:



We prove a lemma stating how this shape combines with itself provided that G contains distinct separation pairs (x, y) and (x', y').

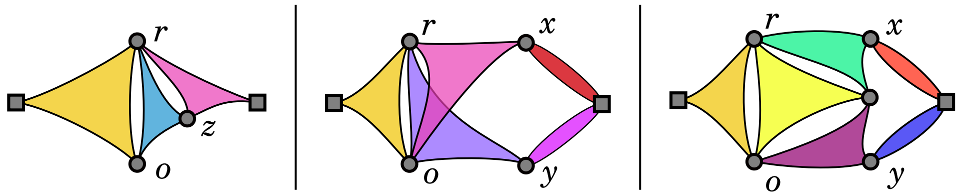
For example this may result in the following combining shape:



In the two sections after the current one, we show how to refine these shapes combining separation pairs. Proposition 5.2.2 is a consequence of this refinement.

5.4 Combining separation pairs on graphs.

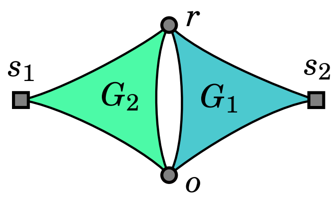
Proposition 5.4.1. *Let G be a hard graph containing two distinct minimal forget separation pairs (x, y) and (r, o). Up-to swapping x and y and r and o, either x is a checkpoint of (G, r), or G has one of the following three shapes:*



where in the left-hand side shape, $z \notin \{x, y\}$ and x and y are in the rightmost pink component.

Observe that only the rightmost two shapes of this lemma combine the shapes of the two separation pairs (x, y) and (r, o).

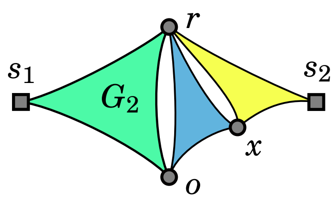
Proof. Since (r, o) is a separation pair, G has the following shape:



In this shape, edges are labelled by the graphs that should substitute the edges to retrieve G.

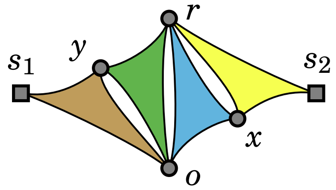
We distinguish several cases depending on where x and y are with respect to this shape. Without loss of generality assume that x is an inner element of G_1.

- $y = r$; it follows from Lemma 4.25.4 that x is a checkpoint between o and s_2 in G_1. Thus G has shape

  and x is a checkpoint in (G, r).

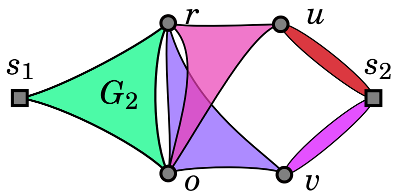
- y is an inner element of G_2; it follows from Lemma 4.25.4 that, up-to swapping r and o, x is a checkpoint between o and s_2 in G_1, and y a checkpoint between s_1 and r i G_2. In other words, G has the following shape:



  and x is a checkpoint in (G, r).

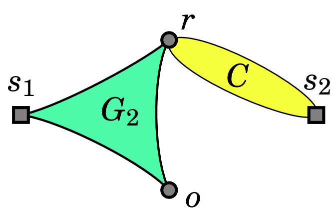
- y is an inner element of G_1; we distinguish two cases depending on whether x and y are in the same prime component of G_1 or not.

  x and y are in different prime components of G_1; let C_x and C_y the prime components of G_1 containing x and y, respectively. As a consequence of Lemma 4.25.4, no other prime component of G_1 is incident to both s_2 and a vertex in $\{r, o\}$, and x (resp. y) is a checkpoint in C_x (resp. C_y) between r and s_2 and between o and s_2. Thus G has the following shape:



  x and y are in the same prime component of G_1; as a consequence of Lemma 4.25.4, no other prime component of G_1 is incident to both s_2 and a vertex in $\{r, o\}$ and r and o are not adjacent to s_2. Let C be the prime component of G_1 containing x and y.

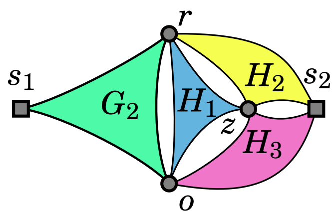
  We prove that C is full. If it is not, then up-to swapping r and o, G has shape:



and r is a anchor in G, a contradiction.

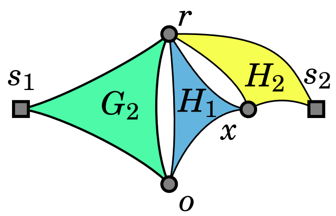
Since C is non-atomic full and prime, it has a forget point z (Proposition 4.4.2) and the series remainder of C at z is empty (Proposition 4.2.4 since r and o are not adjacent to s_2).

Hence G has the following shape:



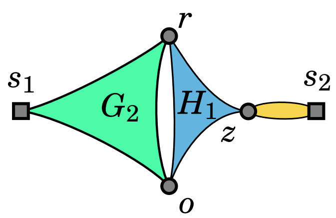
We distinguish several cases depending on where are x and y in this shape.

- $z = x$**;** up to swapping r and o (and hence H_2 and H_3), assume that H_3 does not contain y. As a consequence of Lemma 4.25.4, there does not exist a path in H_3 from o to s_2. Hence G has the following shape:



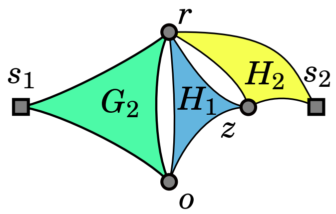
and x is a checkpoint in (G, r).

- x **and** y **are both inner elements of** H_1**;** by Lemma 4.25.4, there is no path in H_2 (resp. H_3) from r (resp. o) to s_2. Hence G has the following shape:



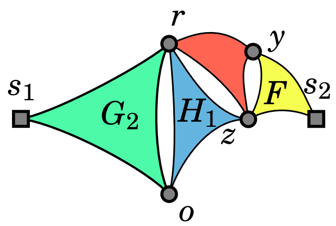
and z is an anchor in G, a contradiction.

- **x and y are both inner elements of H_2;** by Lemma 4.25.4, paths in H_3 from o to s_2 cannot exist. Thus, G has the following shape:

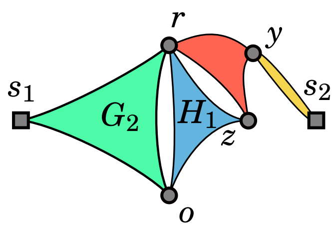


$z \notin \{x, y\}$ and x and y are inner elements in the rightmost component (that is, in H_2).

- **x is an inner element of H_1 but not y;** up to swapping r and o (and hence H_2 and H_3), assume that y is inner in H_2. As a consequence of Lemma 4.25.4, y is a checkpoint from r to s_2 in H_2, and there are no o-s_2 paths in H_3. Hence, G has the following shape:

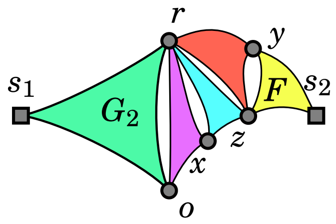


We prove that there exist paths from z to s_2 in F. If not, then G has the following shape:



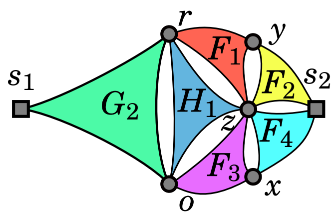
and y is an anchor in G.

As a consequence of Lemma 4.25.4 and since there exists paths from z to s_2 in F, x is a checkpoint between o and z in H_1, and G as the following shape:

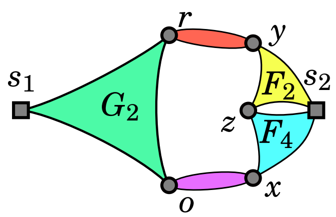


Hence x is a checkpoint in (G, r).

- **x is an inner element in H_3 and y in H_2;** By Lemma 4.25.4, x is a checkpoint between o and s_2 in H_3, and y is a checkpoint between r and s_2 in H_2. Thus, G has the following shape:

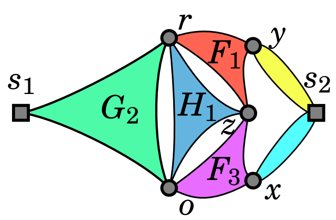


If there are no path from r or o to z in either H_1, F_1, or F_3, then G has the following shape:



and x is a checkpoint in (G, r).

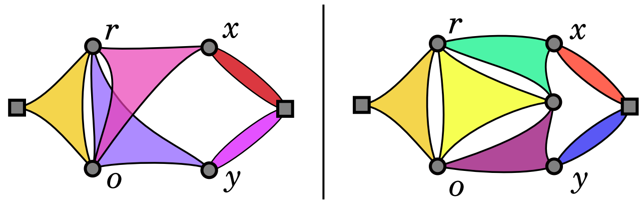
Otherwise, as a consequence of Lemma 4.25.4, there cannot exist paths from z to s_2 in either F_2 or F_4, and G has the following shape:



□

5.5 Combining separation pairs on terms.

Lemma 5.5.1. *Let t be a hard term containing a minimal forget separation pair (x, y). Either t reaches (x, y), or t reaches a minimal forget separation pair (r, o) such that $\mathcal{G}(t)$ has one of the following two shapes:*



Proof. By Proposition 4.26.2, t reaches a minimal separation pair (r, o). If either $r \in \{x, y\}$ or $o \in \{x, y\}$, say $r = x$ without loss of generality, then since y is a checkpoint in $(\mathcal{G}(t), x)$, t reaches x, and since every term reaches its checkpoints, t reaches (x, y).

Otherwise, $\{x, y\}$ and $\{r, o\}$ are disjoint and we proceed by induction on the sum of the orders of the components of $(\mathcal{G}(t), r, o)$ containing x and y, respectively.

We apply Proposition 5.4.1. For the leftmost shape, we proceed recursively: since z is a checkpoint in $(\mathcal{G}(t), r)$, t reaches (r, z). The components of $(\mathcal{G}(t), r, z)$ containing x and y are smaller than in $(\mathcal{G}(t), r, o)$. Hence, we proceed recursively. □

SOURCED MINORS AND K_3

Let t be a hard term of treewidth at most three of S-shape for S one of the two graphs in the statement of Lemma 5.5.1. Observe that (S, x, y) has treewidth four. Since (x, y) is a forget separation pair, $(\mathcal{G}(t), x, y)$, on the other hand, has treewidth at most three. In the next section, we use this fact to refine S and obtain a structural property for hard graphs of treewidth at most three. This refinement uses a characterisation of graphs without a sourced version of the triangle K_3 as a minor. This section is devoted to the proof of this characterisation.

5.6 Sourced minors. A *sourced minor of a sourced simple graph* is a sourced simple graph obtained by a sequence of the following operations: edge and vertex deletions, and edge contractions. A *sourced minor of a graph* is a sourced minor of its footprint.

Proposition 5.6.1. *Let G be a graph. If a sourced simple graph H is a sourced minor of G, then H's skeleton is a minor of G's skeleton.*

Proof. By definition of source minors and Proposition 1.15.1. □

Compared to minors, we use sourced minors to gain "control" on sources.

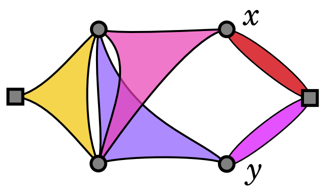
Sourced minors relate to the standard notion of rooted minors. Rooted minors are one of the main tools in the graph minor theory of Robertson and Seymour [32].

Henceforth we write K_k for the sourced simple graph of arity k in which every pair of vertices is an edge.

Proposition 5.6.2. *Given a graph G of H-shape, if G contains K_k as a sourced minor, then so does H.*

Proposition 5.6.3. *Given a graph G and a substitution σ, if, for every integer k and variable a of arity k appearing in G, K_k is a sourced minor of $\sigma(a)$, then the footprint of G is a sourced minor of $\sigma(G)$.*

For example, assume that a graph G has the following H-shape from Proposition 5.4.1:



Let σ be a substitution such that $G \simeq \sigma(H)$. If the hypothesis of Proposition 5.6.3 applies to G and H (for every variable a of H, $\sigma(a)$ contains the complete graph $K_{\mathrm{ar}(a)}$ as a sourced minor), then the footprint of H is a sourced minor of G. Observe that (H, x, y) has treewidth four. Hence, if (x, y) is a forget pair of G, then Proposition 5.6.3 does not apply. If σ is a substitution such that $G \simeq \sigma(H)$, then for some variable a of arity k labelling an edge of H, K_k is not a sourced minor of $\sigma(a)$. Observe that when $k = 2$, this implies that $\sigma(a)$ contains no full prime component.

Next, we see how to deduce a similar fact when $k = 3$.

5.7 K_3 sourced minors and shapes.

Note that graphs excluding K_3 as a sourced minor are not necessarily acyclic, as cycles may occur away from the sources. Hence understanding which graphs exclude K_3 as a sourced minor is different from Proposition 1.15.2.

Lemma 5.7.1. *Let G be an arity three graph and three inner paths P_1, P_2, and P_3, respectively from the first to the second, the second to the third, and the third to the first sources of G.*

If no vertex is common to all three paths then G admits K_3 as a sourced minor.

Proof. Endpoints and vertices of paths are preserved by taking the footprint of a graph: the lemma holds for a graph whenever it holds for its footprint. Hence we may assume that G is a sourced simple graph.

We write s_1, s_2, s_3 for the three sources of G.

We reason by induction on $|P_1| + |P_2| + |P_3|$, the sum of the lengths of the three paths.

If P_1, P_2, and P_3 are all of length one then they are reduced to edges connecting pairs of sources. Said otherwise, G contains K_3 as a subgraph, and by extension as a minor.

Otherwise, at least one of the paths P_i, say P_1, has an inner vertex. We note e for the first edge and x for the first inner vertex of P_1.

If x does not appear in either P_2 or P_3 then we contract e in G to get G/e. This shortens P_1, while retaining the hypothesis: we conclude by induction on G/e.

Otherwise, x appears in, say, P_3. We distinguish two cases:

1. **x is not the last inner vertex of P_3;** in this case, we replace P_3 by the shorter path $P_3 x s_1$. Path P_1, P_2, and $P_3 x s_1$, still have no vertex common to all three: so we conclude by induction hypothesis.

2. **x is the last inner vertex of P_3;** the last edge of P_3 is $x s_1 = e$. We contract e and conclude by induction hypothesis on the resulting minor. (x not being in P_2 guarantees that contracting e does not modify P_2.) □

A *sourced star* is a sourced simple graph whose vertices are all sources, and whose skeleton is a star.

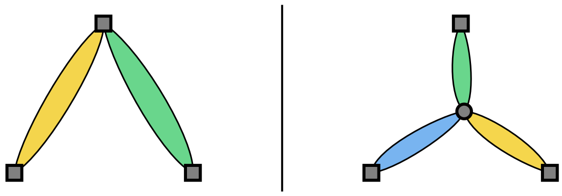
Proposition 5.7.2. *Let G be a full prime graph and s a source of G. The sourced star with vertex set $V(G)$ and centre s is a sourced minor of G.*

Proof. If G has no inner vertex then it must be atomic, it admits as a sourced minor the clique on its k sources, and, in particular, the graph from the statement.

Otherwise, starting from the footprint of G, contract all inner edges connecting non-sourced vertices. We end up with a single inner vertex, denoted by x, which is connected to all sources (thanks to G being full prime). Finally, contract the edge connecting x and s. □

The next result was first proved in [50], where it is formulated for rooted minors. Their proof proceed by induction and follows the principles described in §1.19. Ours proceed by case analysis on the full prime components of the graph.

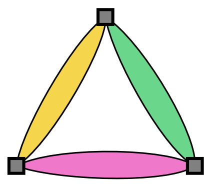
Proposition 5.7.3. *Every graph of arity three has either K_3 as a sourced minor, or one of the following two shapes:*



Moreover, these two shapes do not have K_3 as a sourced minor.

Proof. Let G be a graph of arity three without K_3 as a sourced minor.

If G has no full prime components, then it has the following shape:



Since K_3 is not a sourced minor of G, some edge of this shape has no full prime components in G. Thus G has the left-hand side shape in the statement.

If G contains two or more full prime components, then G has K_3 as a sourced minor as a consequence of Proposition 5.7.2 applied to two distinct full prime components and sources of G. This is a contradiction.

Assume that G contains exactly one full prime component C. If a reduced component D of G had arity two, then as a consequence of Proposition 5.7.2 applied to C and the source of G which is not incident to D, K_3 is a sourced minor of G.

Assume that G contains exactly one full prime component C and furthermore that all other reduced components of G have arity at most one.

We prove that there exists a vertex x of C which is a checkpoint between all pairs of sources: it follows that G has the right-hand side shape from the statement.

Write s_1, s_2, and s_3 for the sources of G. Let P_1, P_2, and P_3 be three paths in C respectively from s_1 to s_2, s_2 to s_3, and s_3 to s_1, each of minimal length.

By Lemma 5.7.1, there is a vertex x common to these three paths.

Since segments such as P_1x and $\overline{P_3}x$ share their endpoints, they must have equal length: otherwise one could be replaced by the other to obtain a shorter triple of paths.

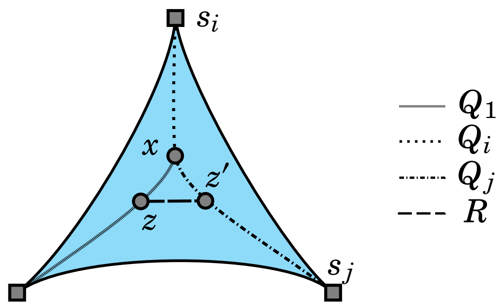
Without loss of generality, we chose them to be equal. Accordingly, we write Q_1 for $P_1x = \overline{P_3}x$, Q_2 for $P_2x = \overline{P_1}x$, and Q_3 for $P_3x = \overline{P_2}x$. The path Q_i is from s_i to x.

The paths Q_i are pairwise inner disjoint. Indeed, would Q_1 and Q_2 share an inner vertex (for instance), then $P_1 = Q_1\overline{Q_2}$ would visit that vertex twice.

We finally prove that all inner paths between any two sources must contain x. Let Q be an inner path between two sources, say s_1 and s_2 without loss of generality, and assume by contradiction that Q avoids x.

By Lemma 5.7.1, there is a vertex y shared by Q, P_2, and P_3. Since Q_1 and Q_2 are inner disjoint, y must belong to Q_3.

The path Q starts at s_1, ends at s_2, and meets Q_3 at y. It cannot always remain on Q_1 as it does not contain x. Let z be the last vertex of Q in common with Q_1, and z' be the first vertex after z on Q that meets Q_j for $j \in \{2,3\}$. Note that the inner part of $R = zQz'$ is disjoint from all Q_is. As z and z' cannot be x, we are in the following situation:

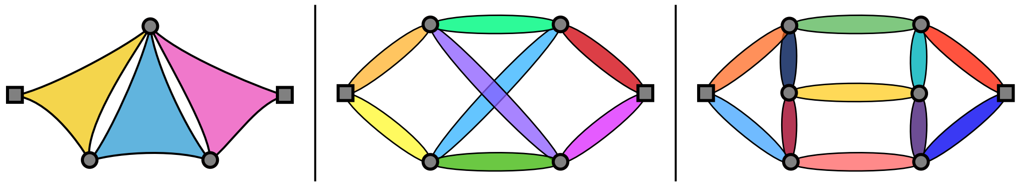


It follows easily, using the fact that most drawn paths are disjoint, that K_3 is a sourced minor of G, a contradiction. □

HARD GRAPHS OF TREEWIDTH AT MOST THREE

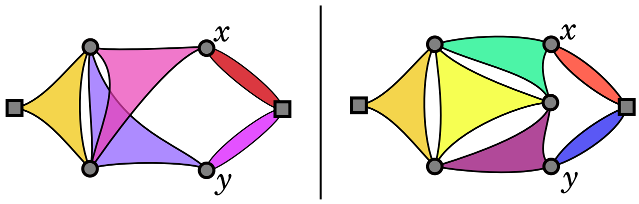
5.8 Hard graphs of treewidth at most three: structure.

Theorem 5.8.1. *Every hard graph of treewidth at most three has one of the following three shapes in which separation pairs are all minimal forget pairs:*



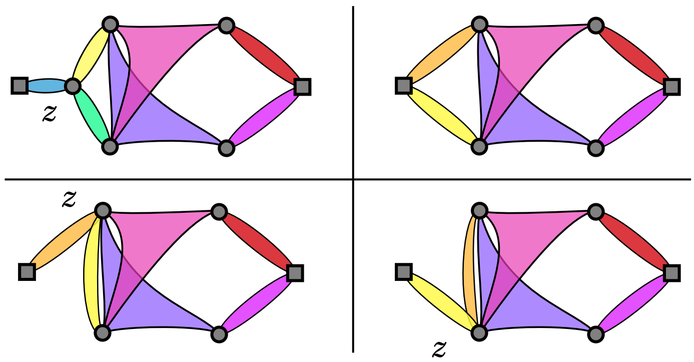
In the two rightmost shapes, every graph which substitute an edge to retrieve G contains full prime components.

Proof sketch. Let G be a hard graph of treewidth at most three. As a consequence of Lemma 5.5.1 we know that either G has the left hand side shape of the statement, or one of the following two shapes:



Write H to denote one of these two shapes. The graph (H, x, y) has treewidth four. By Proposition 5.6.3 we know that one of the graphs substituting some edge e of H to retrieve G does not contain the clique as a sourced minor.

Assume that H is the leftmost shape above, and e the leftmost edge of H. By Proposition 5.7.3, G must have one of the following four shapes:



In three of these shapes, the vertex z is an anchor of G, a contradiction. There remains only one shape to study.

Repeating this arguments proves that the only two possibilities are the ones in the statement.

An interactive and complete version of this proof can be found at [31]. A complete proof can also be found in [15]. □

5.9 Hard terms of treewidth at most three. Theorem 5.2.1 is a consequence of the following.

Proof of Proposition 5.2.2. By combining Lemma 5.5.1 with the proof of Theorem 5.8.1, applying Proposition 4.17.1 in one case, and FX or FD in the others. □

Hence we have obtained, through results in the previous and current chapters, a positive answer to the treewidth axiomatisation conjecture for graphs of treewidth at most three.

As a consequence, we retrieve, in the next paragraph, the structural characterisation of graphs of treewidth at most two by avoiding K_4 as a minor.

5.10 Excluded minors of treewidth at most two graphs. According to Theorem 1.15.3, simple graphs of treewidth at most k are characterised by a finite list of minimal excluded minors. We saw in Proposition 1.15.6 that K_{k+2} belongs to this list.

A *sourced minimal excluded minor* of treewidth at most k is one of the minors described in Theorem 1.15.3 in which the vertices of a maximal clique (with respect to inclusion) are promoted to sources.

Proposition 5.10.1. *The sourced minimal excluded minors of graphs of treewidth at most k which are distinct from K_{k+2} are hard graphs of treewidth at most $k + 1$.*

Proof. Let G be a sourced minimal excluded minor of graphs of treewidth at most k distinct from K_{k+2}. By minimality, G does not contain K_{k+2} as a sourced minor. Hence the arity of G is at most $k + 1$. If G is not prime or full, the we can write G as a parallel composition or as a lift of some of its minors. By Proposition 1.16.3 that would imply that G has treewidth at most k. Similarly G cannot be atomic.

Hence G is non-atomic full prime. Any non-atomic full prime graph of arity $k + 1$ contain K_{k+2} as a source minor. Hence G has arity at most k.

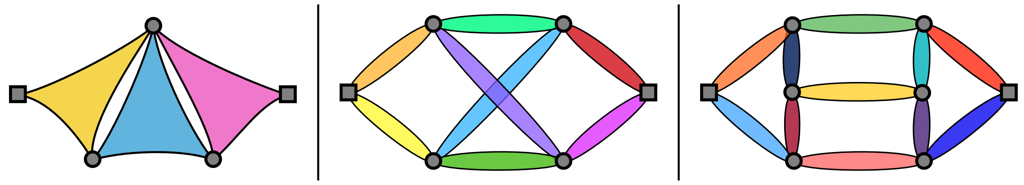
As the sources of G are a maximal clique in G's skeleton, G cannot contain a clique point.

If G contained an anchor x, then we can parse on x to prove that x is a k-forget point of G, a contradiction.

Hence G is hard. □

Theorem 5.10.2. *A graph has treewidth at most two if and only if it does not contain K_4 as a minor.*

Proof. For the sake of contradiction, let G be a sourced minimal excluded minor of treewidth at most two which is distinct from K_4. By Proposition 5.10.1 It is a hard graph of treewidth at most three. By Theorem 5.8.1, G has one of the following three shapes:



where in the two rightmost shapes, every graph which substitute an edge to retrieve G contains full prime components. In the case of one of the two rightmost shapes, G has the shape as a minor. Both shapes having K_4 as a minor, this is a contradiction.

In the case of the leftmost shape, having that G is hard ensures the existence of some paths in G: the leftmost component, for example, contains inner paths from the source it is incident with to each of the two inner vertices it is incident with.

Such conditions then ensure that G has K_4 as a minor, a contradiction. □

CONCLUSION

Since the beginning of Chapter 4, we have provided different technical, completeness, and incompleteness results for several axiom sets. A summary of all these results can be found in Figure 4. In Chapter 7, we discuss attempts towards a generalisation of our proof of Theorem 5.2.1 to graphs of treewidth at most four.

These attempts rely on the two paths theorem; this is the topic of the next chapter.

6

Two-paths theorem

6.1 Introduction. The general problem of finding disjoint paths between fixed source and target vertices in a graph has been extensively studied in the last fifty years. It is an important tool in graph minor theory, where it is used to find minors in graphs and understand the structure of minor-excluding classes of graphs [32].

In the third section of this chapter we prove the two-paths theorem (§6.11 and §6.12), a structural characterisation of graphs G in which there exists four distinct vertices s_1, s_2, t_1, t_2 without disjoint s_1-t_1 and s_2-t_2 paths. This theorem uses *webs*: graphs built by filling with cliques the inner faces of *ribs*—planar graphs whose outer face is bounded by a cycle of length four with inner faces being triangles and triangles being faces.

In a similar manner as in [34, 24], our proof relies on a generalised statement that we introduce in the first section (§6.2 to §6.4), and on an operational characterisation of a generalised notion of web (§6.5 to §6.10). This characterisation is new, even though the proofs in [34, 24] clearly hint at it.

In a last section, we describe a novel algorithm to solve the two disjoint paths problem (§6.13 to §6.17), which consists of deciding, given s_1, s_2, t_1, t_2 four distinct vertices in a graph G, whether G contains two disjoint paths, one from s_1 to t_1, and the other from s_2 to t_2.

In this chapter we only consider simple graphs. We work with planar graphs. A general introduction to planarity in graph theory is found in [13, Chapter 4].

GENERALISED WEBS AND TWO PATHS THEOREM

6.2 The standard two paths theorem. A graph is called 2-*linked* if for all pairwise distinct vertices s_1, s_2, t_1, t_2, there exists two disjoint paths, respectively from s_1 to t_1 and from s_2 to t_2. Such a pair of paths is called an (s_1, s_2, t_1, t_2)-*linkage*.

A *rib* is a planar graph such that in one of its plane embeddings, the outer face is a cycle of length four, all inner faces are triangles, and all triangles are inner faces. A *web* is a graph constructed from a rib R by adding one clique K_T per triangle T of R, making vertices of K_T adjacent to exactly the vertices of T. The outer face of a rib or of a web is called its *frame*.

Theorem 6.2.1 (Two paths theorem [43, 38]). *The webs are exactly the edge-maximal graphs which are not* 2*-linked.*

Given two vertices x and y of a graph G such that xy is not an edge of G, we refer to $G + xy$ as an *edge extension* of G.

This theorem can be shown to be equivalent to the following statement: let $G = (V, E)$ be a graph and let s_1, s_2, t_1, t_2 be four pairwise distinct vertices of G. The following statements are equivalent:

1. G is a web with frame $s_1s_2t_1t_2s_1$,
2. every edge extension of G contains an (s_1, s_2, t_1, t_2)-linkage.

To prove the two paths theorem, Thomassen [43] relies on Kuratowski's theorem after reducing the problem along small separators. Independently, Seymour [38] sketched a proof reducing the problem to the 2-connected case before a clever combinatorial analysis of the graph's separators. Hence both proofs are connectivity-oriented (as described in §1.19). Moreover, both use Menger's theorem.

A proof is also provided in [34, (2.4)] and later reworked in [24, Appendix A]. It also relies on connectivity-reduction via Menger's theorem, afterwhat the method is similar to ours.

Our proof departs from such connectivity reduction strategies and uses neither Kuratowski's nor Menger's theorems. Instead we use graph operations generating webs inspired by parallel compositions as defined in §1.16.

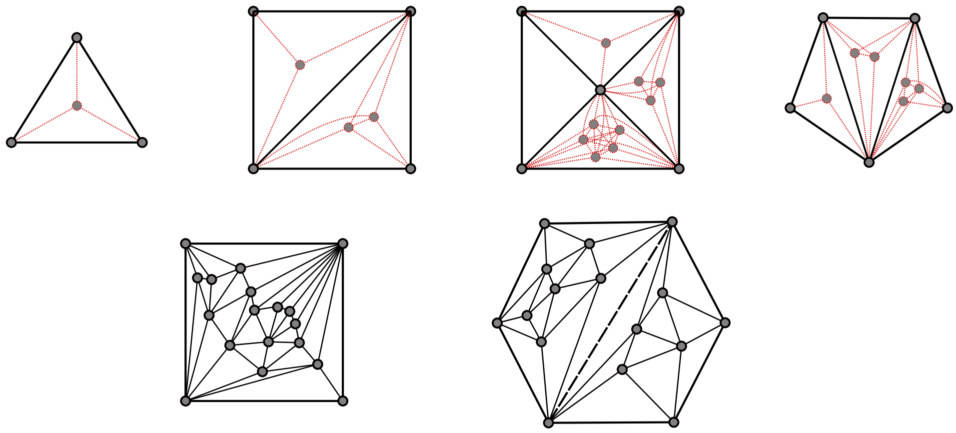


Figure 6.1: Examples of webs and ribs; dotted red vertices and edges are clique vertices and edges, while plain black ones are rib vertices and edges. The black dashed edge in the bottom right web is discussed in §6.5. The ribs of the webs on the top part are the only ribs of order three, four, and five.

The standard version of webs and the two paths theorem are not flexible enough to carry such a study. We start by generalising them.

This generalisation of webs is first introduced in [34] in the form of a *rural societies*.

6.3 Generalised webs.

Definition 6.3.1. Given $k \geq 3$, a *k-rib* is a planar graph such that in one of its plane embeddings, the outer face is a k-cycle called its *frame*, all inner faces are triangles, and all triangles are inner faces.

A *k-web* is a graph constructed from a k-rib R by adding one clique K_T per triangle T of R, making vertices of K_T adjacent to exactly the vertices of T.

In the sequel, when the size of the frame is not relevant, we often simply call ribs and webs the graphs from this generalised definition. Ribs are particular cases of graphs called “*near-triangulations of the plane*” in the literature (see [46]), and webs are exactly the 4-webs.

In Chapter 7, we show that there exists ribs of arbitrary treewidth, simply by adding edges to the *k×k grid*. It is well-known that the latter has treewidth k.

Some examples of webs and ribs are depicted in Figure 6.1.

Proposition 6.3.2. *The* 3*-ribs are all triangles; the* 3*-webs are the cliques of order at least three.*

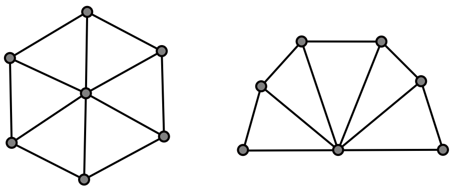


Figure 6.2: The graphs W_7 and F_7, depicted on the left and right-hand sides, respectively.

To understand the shapes of k-ribs, and by extension of k-webs, we describe the subgraphs induced by neighbourhoods in ribs.

In a graph, a *universal vertex* is a vertex adjacent to all other vertices.

For example, the centre of a star graph is universal.

We define the following standard graph families:

- the *wheel graph* with $n \geq 4$ vertices is the graph W_n obtained from the cycle of length $n - 1$ by addition of a universal vertex;
- the *fan graph* with $n \geq 3$ vertices is the graph F_n obtained from the path of length $n - 2$ by addition of a universal vertex.

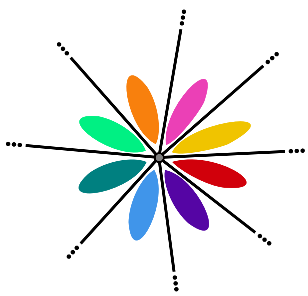
Examples of a wheel and a fan graph are shown in Figure 6.2.

Observe that if e is an edge in a wheel graph that is not incident to the universal vertex, then $F_n \simeq W_n - e$.

Proposition 6.3.3. *Let x be a vertex in a rib R with frame C. The subgraph of R induced by x and $N_R(x)$ is:*

- *a fan graph when x is in C, and*
- *a wheel graph otherwise.*

Proof. Consider a plane embedding of R in which C bounds the outer face, all inner faces are triangles, and all triangles are inner faces. Let H be the subgraph of R induced by x and its neighbours. In the embedding of R, the edges going out from x form a star whose centre is x:



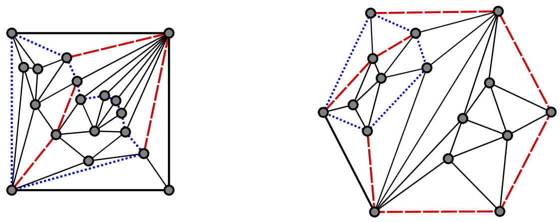


Figure 6.3: Two examples of crossings (the red dashed paths) for some cycles (the blue dotted cycles).

Every consecutive pair of edges bounds a common face which is represented by a coloured area in the drawing above. Either all such faces are inner, meaning are triangles, and H is a wheel graph (and x cannot bound the outer face, meaning $x \notin V(C)$), or some of these faces are the outer face. The outer face being bounded by a cycle C, at most two edges incident to x may bound the outer face. In this case $x \in V(C)$ and H is a fan graph. □

6.4 Generalised two paths theorem. Let $T = (x_1, \ldots, x_k)$ be a tuple of at least three pairwise distinct vertices in a graph G. A *crossing* of T is a pair of disjoint T-paths, from x_i to x_j and x_r to x_s, respectively, such that $i < r < j < s$. The tuple T is *crossed* if it admits a crossing; it is *crossless* otherwise. The graph G is *maximally T-crossless* when T is crossless and in any edge extension of G, T crossed.

Two examples of crossings are given in Figure 6.3; they are crossings in the bottom ribs of Figure 6.1 but for tuples distinct from their frames.

Observe that a graph is 2-linked precisely when all tuples of four distinct vertices are crossed.

Theorem 6.4.1 (Generalised two paths theorem)**.** *Let G be a graph with a cycle C. The following are equivalent:*

- *G is a web with frame C,*
- *G is maximally C-crossless.*

Theorem 6.2.1 is the particular case where $|C| = 4$. In the case where $|C| = 3$, C cannot be crossed, thus Theorem 6.4.1 holds since 3-webs are cliques by Proposition 6.3.2.

In [34, (2.4)] and [24, Appendix A], the authors prove a statement that can easily be shown equivalent to:

1. frames of k-webs are not crossed,

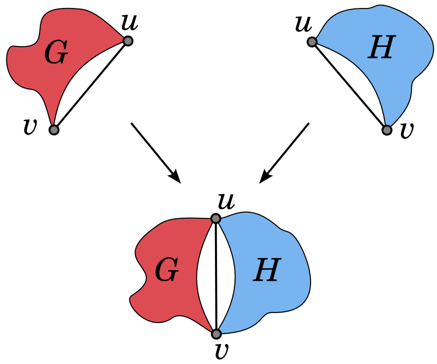


Figure 6.4: The bottom part shows the edge compositions of the graphs G and H on the top part, along uv.

2. each graph G with a cycle C which is not C-crossed is a subgraph of a k-web.

The maximality of k-webs with regards to frames being crossed seems not to be directly deducable from [34, 24].

The motivation for our generalisation from 4-webs to k-webs is that these can be characterised inductively from cliques by successive *parallel compositions*. Webs are simple graphs and not sourced graphs. Hence parallel compositions for webs require a different definition than that of §1.16. The corresponding operation is called *web composition*.

WEB (DE)COMPOSITIONS

6.5 Edge composition. The graph operation consisting in taking the disjoint union of two graphs and then identifying two edges e and e', one from each graph, is called *edge composition along e and e'*.

Edge compositions are illustrated in Figure 6.4.

Edge compositions of webs along edges of their frames always produce webs: plane embeddings can be composed in this way, all triangles remain faces and all inner faces remain triangles. If the identified edges are denoted $e = xy$ and $e' = x'y'$ and the frames of the composed webs are respectively Pxy and $P'x'y'$, then the resulting web has frame $P\overline{P'}$.

An example is the rib in the bottom right corner of Figure 6.1. It can be obtained by edge composition, the black dashed edge resulting from the edge identification. On the other hand, the rib in the bottom left corner of this figure cannot be expressed by an edge composition of webs.

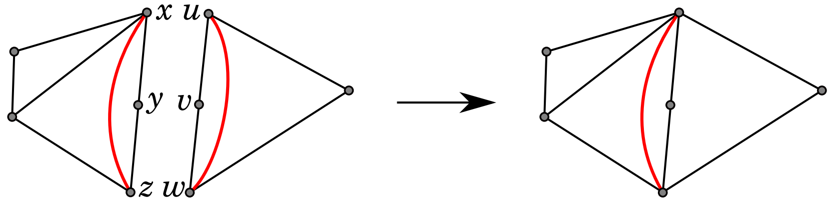


Figure 6.5: The right-hand side shows the path composition of the two ribs on the left-hand side along the paths xyz and uvw. This graph is not a rib. Indeed, the dashed red edge, resulting from the identification of xz and uw, is shared by three triangles; thus, at least one of these triangles is not a face in a plane embedding of the path composition.

6.6 Path composition. To decompose webs in general (except for the base case of 3-webs, i.e., cliques), we generalise edge composition to identify subpaths of frames instead of single edges.

If $P = x_1 \dots x_l$ and $P' = x'_1 \dots x'_l$ are non-empty paths of length l of two graphs G and H, respectively, we define the *path composition* of G with H along P and P' as

$$(G \uplus H)/x_1x'_1/\dots/x_lx'_l.$$

When $P = P'$ we simply talk of *path composition along* P.

In the particular case where P and P' are reduced to one edge each, that is $l = 1$, the notion coincides with edge composition.

With terminology and notations from §1.16 and §4.4, the path composition along P and P' of G and H is:

$$\mathrm{f}^l((G, x_1, \dots, x_l) \parallel (H, x'_1, \dots, x'_l)).$$

Unfortunately, path compositions of webs along subpaths of their frames are not always webs (although this is true for edge compositions, as explained above). A counter-example is given in Figure 6.5; to fix this issue, we have to restrict to induced subpaths of the frames.

6.7 Web composition. Let G and H be two webs and P and P' subpaths of their respective frames of the same length. If P and P' are induced paths in G and H, respectively, we call *web composition* of G with H along P and P' the path composition of G with H along P and P'.

Webs are stable under web compositions:

Proposition 6.7.1. *The web composition along P and P' of two webs G and H with respective frames P_GP and P_HP', is a web with frame $P_G\overline{P_H}$.*

Proof. Let R_G and R_H be the ribs of G and H, respectively, and R the web composition of R_G and R_H along P and P'. We prove that R is a rib.

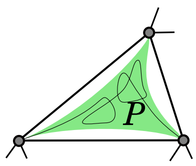
Any planar graph admits a plane embedding for which the outer face is sent exactly onto a given prescribed polygon of the plane (see [44]).

Consider plane embeddings of R_G and R_H for which the outer faces are sent onto two polygons whose intersection is precisely P and P'. As P and P' are induced, there are no parallel edges in the union of the plane embeddings of R_G and R_H. Hence, this union is a plane embedding of R.

As all inner faces of the embedding of R_G or R_H are triangles, so are inner faces of R. As P and P' are induced, a triangle of R is either a triangle of R_G or of R_H, but not of both. Hence every triangle of R is a face in this embedding. So R is a rib.

Adding back the clique vertices of G and H on their respective triangles gives a web, proving that the web composition of G with H along P and P' is a web with rib R and frame $P_G P_H$. □

6.8 Crossings in webs. An immediate yet important observation is that in a web with rib R, the intersection of an R-path with R yields an induced path. Indeed, if such a path P uses a clique vertex u, then we can consider the longest subpath Q of P containing u, and whose inner vertices are all clique vertices. The endpoints of Q are on a triangle T of the rib. In particular Q can be replaced in P by an edge of T:



In this figure the green triangular shape in the centre of the triangle represents all the clique vertices of T.

Conversely, induced paths whose endpoints are on the rib are entirely on the rib.

Thomassen's argument [43] for proving that frames of 4-webs are not 2-linked, generalises to n-webs:

Proposition 6.8.1. *Frames of webs are crossless.*

Proof. Suppose by contradiction that the frame of a web is crossed. The union of the frame and the paths from a crossing yields a topological K_4.

By intersecting paths with the rib, we can assume a topological K_4 in the rib.

In a planar embedding of the rib in which the frame bounds the outer face, this provides an outerplanar plane embedding of K_4, a contradiction. □

6.9 Edge decompositions. We prove a necessary and sufficient condition for a web to be expressible as the edge composition of two webs. It relates to the C-connectivity of vertices in webs of frame C.

Compared with terminology from Chapter 4, with (G, C) the graph G with sources the vertices of its frame C, two vertices x and y are C-connected if and only if there exists an inner path in (G, C) from x to y. Hence, C-connectivity is related to the full prime components of (G, C)

Lemma 6.9.1. *Let x be a vertex in a web G with frame C such that $|C| \geq 4$. There exists a C-path P between non-consecutive vertices c_x and c'_x of C such that x is C-connected to all vertices of P.*

Proof. It is sufficient to prove the statement for ribs. Indeed, except perhaps for its endpoints, we have seen that induced paths in webs are paths in their ribs. Hence, two rib vertices are C-connected in G if and only if they are C-connected in the rib of G; two clique vertices are C-connected in G if and only if two of their rib neighbours are C-connected in the rib of G.

When x is a vertex in C, by Proposition 6.3.3, the neighbourhood of x induces a fan graph F. Write P for the path in $F - x$ whose endpoints are the predecessor and successor of x on C. If P is a C-path, then we take the endpoints of P for c_x and c'_x; P is a C-path from c_x to c'_x whose vertices are all C-connected to x. If P is not a C-path, then P contains a vertex y of C which is adjacent to x. Since y is inner in P, x and y are non-consecutive on C. We take x and y for c_x and c'_x, and $Q = c_x c'_x$. The path Q is a C-path from c_x to c'_x whose vertices are all C-connected to x.

Assume that x is not a vertex of C. Observe that ribs are connected, and consider an induced C-path P from x to a vertex c of C. By Proposition 6.3.3, the neighbourhood of c induces a fan graph in G. Hence, this neighbourhood provides paths from the first vertex u of P which is on F, to the two neighbours d and d' of c on C, and these two paths Q and R are disjoint except possibly for u.

For c_x and c'_x we take the first vertices of Q and R which are on C, respectively (see Figure 6.6 for an illustration when c_x and c'_x are d and d').

All vertices of Q and R are neighbours of c. For the sake of contradiction, suppose that c_x and c'_x are consecutive on C; $c_x c'_x c c_x$ is a triangle of R, and hence an inner face. The edge cc_x bounds at least three faces in the plane embedding of R in which the outer face is bounded by C, all inner faces are

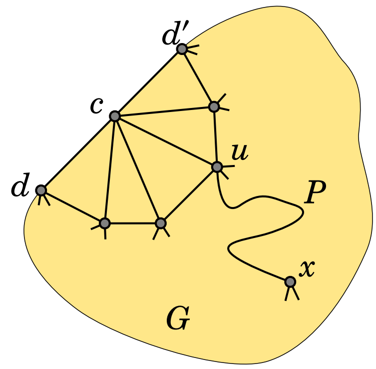


Figure 6.6: The case where $c_x = d$ and $c'_x = d'$ in the proof of Lemma 6.9.1. The coloured area is G, with C being the outer cycle.

triangles and all triangles inner faces. In a plane embedding of a planar graph, an edge may bound at most two faces, a contradiction.

The paths PuQ and PuR are C-paths from x to c_x and to c'_x, respectively. They induce a C-path from c_x to c'_x whose vertices are all C-connected to x. □

Thanks to the previous lemma, we can prove the following characterisation. The second and third items in the proposition are refinements of the first one, which are useful in the sequel.

Proposition 6.9.2. *Let G be a web with frame C. The following holds:*

1. *G is the edge composition of two webs if and only if there is a pair of vertices of G which are not C-connected.*

2. *for all vertices x, y, either x is C-connected to y, or there exists a pair of vertices $c, c' \in C - x - y$ separating x from y and such that G is the edge composition of two webs along cc',*

3. *for every vertex x and every subpath P of C, either x is C-connected to P, or there are vertices c, c' of $C - P - x$ separating x from P and such that G is the edge composition of two webs along cc'.*

Proof. The first item of the statement is a direct consequence of the second one, which we prove now.

Let G be a web with frame C and two vertices x and y which are not C-connected.

By Proposition 6.3.2, all pairs of vertices in 3-ribs with frames triangles T are T-connected. Hence we assume $|C| \geq 4$.

By Lemma 6.9.1, x is C-connected to at least two distinct non-consecutive vertices c_x and c'_x of C. Write Q for a C-path from c_x to c'_x containing vertices all C-connected to x. Let P and P' be the two paths from c_x to c'_x such that $P\overline{P'} = C$.

We prove that the vertices of C which are C-connected to y are either all on P or all on P'. For the sake of contradiction, suppose that y is C-connected to inner vertices c_y and c'_y of P and P', respectively. Define a C-path R from P to P' as follows:

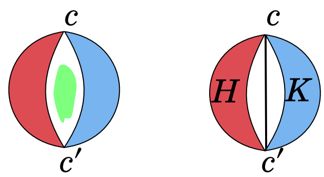
- when y is not on C, R is a c_y-c'_y path containing only vertices C-connected to y,
- when y is on C, we assume without loss of generality that y is an inner vertex of P. For R we take a y-c'_y path.

Since x and y are not C-connected, Q and R are disjoint and (Q,R) is a crossing of C, a contradiction.

Hence, we assume that the vertices of C to which y is C-connected are all on P, and without loss of generality, we assume that P is of minimum length such that its endpoints c_x and c'_x are C-connected to x and all vertices C-connected to y are on P.

We chose $c = c_x$ and $c' = c'_x$. We prove that $\{c, c'\}$ separates $\mathring{P}$ from $\mathring{P}'$. For the sake of contradiction, let R be a C-path from $\mathring{P}$ to $\mathring{P}'$. Since x is not C-connected to $\mathring{P}$, the vertices of R are not C-connected to x either. Recall that Q is a C-path from c to c' whose vertices are all C-connected to x. The pair (Q,R) is a crossing of C, a contradiction.

Because $\{c, c'\}$ separates $\mathring{P}$ from $\mathring{P}'$, a plane embedding of the rib of G in which the outer face is bounded by C, all inner faces are triangles and all triangles are inner faces, has the left-hand side shape below:



Without being too formal on the manipulation of such embeddings, the green area must correspond to an inner face of the embedding, meaning a triangle. This face being incident to both c and c', that implies that cc' is an edge of G. The fact that the rib of G is given by an edge composition of two ribs along cc' is then direct by looking at the inner faces and triangles of H and K. Hence, G is an edge composition of H and K in which the clique vertices are restored.

We prove that $\{c, c'\}$ separates x from y. For the sake of contradiction, let R be an x-y path in $G-c-c'$. Since x and y are not C connected, R contains at least one vertex of C. Write $c_1, \ldots, c_n$ for the vertices of R which are vertices of C: $R = Rc_1Rc_2 \ldots c_nR$. Since x and y are not C-connected to $\mathring{P}$ and $\mathring{P}'$, respectively, we have that c_1 and c_n are on $\mathring{P}'$ and $\mathring{P}$, respectively. Hence, there exists an index i such that c_i is on $\mathring{P}'$ and c_{i+1} on $\mathring{P}$. The path c_iRc_{i+1} is a C-path from $\mathring{P}'$ to $\mathring{P}$ in $G - c - c'$, a contradiction.

The last item is proved along the same lines, replacing y with a subpath of C. □

Corollary 6.9.3. *In a web whose frame C has length at least four, there is a C-path whose endpoints are non-consecutive vertices of C.*

Proof. As $|C| \geq 4$, consider two non-consecutive vertices c, c' of C.

If c is C-connected to c' then this provides a C-path whose endpoints are non-consecutive vertices of C. Otherwise, by Proposition 6.9.2, the web is given as an edge composition along some edge uu' between nonconsecutive vertices of C. This edge is a C-path whose endpoints are non-consecutive vertices. □

6.10 Web decompositions. While the bottom-left web in Figure 6.1 does not arise as an edge composition, it can be expressed as a web composition. In fact, we have the following decomposition result:

Proposition 6.10.1. *Every web whose frame has length at least four is a web composition of two webs.*

Proof. Let G be a web with rib R and frame C with $|C| \geq 4$. By Corollary 6.9.3 there is a C-path P whose endpoints are non-consecutive vertices of C. We assume without loss of generality that P is an induced path, i.e., a path in R. Considering a plane embedding of the latter, the path P separates R into two planar graphs whose outer faces are bounded by the two cycles distinct from C and induced by C and P, all their faces are faces of R and hence triangles, and all their triangles are triangles of R and hence faces: those two planar graphs are ribs. Adding back the clique vertices of G on their respective triangles, the resulting pair of webs generate G by taking their web composition along P. □

Altogether, Propositions 6.3.2, 6.7.1, and 6.10.1 show that the webs are precisely those graphs obtained by repeated web compositions, starting from cliques of order at least three (whose frames are any triangle). Note that such decompositions are not unique in general.

PROVING THE TWO PATHS THEOREM

We now prove the generalised two paths theorem. We have already seen that web frames are crossless (Proposition 6.8.1); we first show that the webs are maximally so. The proof of this result is later reused to exhibit crossings in our presentation of the algorithm for the two disjoint paths problem (§6.16).

6.11 Web maximality.

Lemma 6.11.1. *The frame of a web G is crossed in all edge extensions of G.*

Proof. We reason by induction on the order of G. Let C be the frame of G. Since 3-webs are cliques (Proposition 6.3.2) they cannot be extended with a new edge. We may thus assume $|C| \geq 4$, and hence by Proposition 6.10.1, that G is the web composition of two webs H and K along some induced C-path P of G. Let C_H and C_K be the frames of H and K, respectively. By induction hypothesis, C_H (resp. C_K) is crossed in any edge extension of H (resp. K).

Let $e = xy$ be an edge which is not in G. We prove that C is crossed in $G + e$. To build such a crossing we will rely on the following claim:

Claim 6.11.2. Every vertex of H is C_H-connected to $C_H - P$ in H, and similarly in K.

Proof. As P is induced in both H and K, these webs cannot be expressed as some edge composition along some edge between non-consecutive vertices of P. The claim follows by the third item of Proposition 6.9.2 ◁

Assume that x and y are in $H-P$ and $K-P$, respectively. As x is C_H-connected to $C_H - P$ in H and y is C_K-connected to $C_K - P$ in K, we can assume to have an x-$(C_H - P)$ (resp. y-$(C_K - P)$) path P_x in H (resp. P_y in K) disjoint from P. The pair $(P_x e P_y, P)$ is a crossing of C in $G + e$.

Otherwise e has both its endpoints in either H or K, say in H. By induction hypothesis, let (P_1, P_2) be a crossing of C_H in $H + e$. We reason by case analysis depending on which endpoints of P_1 and P_2 are on P. Up-to symmetries between P_1 and P_2, there are five cases, represented in Figure 6.7 with the shape of the crossing of C we build in $G+e$. The first three cases, in which both P_1 and P_2 have at least one end not on P are self-explanatory and just require the use of P to extend P_1 and/or P_2 to form a crossing of C in $G + e$.

In the fourth case of Figure 6.7, if the end of P_1 which is on P is denoted by z, then, since z is C_K-connected to $C_K - P$ in K, there exists a z-$(C_K - P)$ path P'. Without loss of generality, assume that P_2 is from x to y; the pair (PxP_2yP, P_1zP') is a crossing of C in $G + e$.

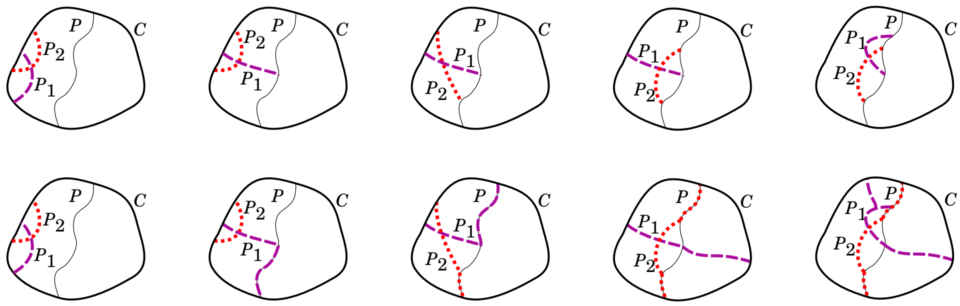


Figure 6.7: The five subcases in the proof of Lemma 6.11.1 (top), with crossings of the outer cycle we deduce in each case (bottom).

In the fifth case, we first observe that P_1 and P_2 cannot be reduced to edges as P is induced. All their inner vertices being C_H-connected to $C_H - P$, consider an associated path P'. Up-to replacing P' by its subpath starting at the last vertex of either P_1 or P_2, we get, say, a $(C_H - P)$-$\mathring{P}_1$ path inner disjoint from P_1, P_2, and C_H. Denote it by P' with u its end on P_1. If z is the end of P_1 that is surrounded by P_2's endpoints on P, then just as above we get a z-$(C_K - P)$ path P'' in K that is inner disjoint with C_K. Without loss of generality, assume that P_2 is a path from x to y; $(P'uP_1zP'', PxP_2yP)$ is a crossing of C in $G + e$. □

6.12 The two paths theorem.

We finally prove the direct implication of the two paths theorem, which is the difficult one in the literature.

Lemma 6.12.1. *Let G be a graph with a cycle C. If G is maximally C-crossless, then G is a web with frame C.*

Proof. Let G be a maximally C-crossless graph for a cycle C. Edges of C cannot be used in crossings. By maximality of G, C is a subgraph of G. We proceed by induction on $|G|$.

We perform case analysis on whether all pairs of distinct vertices of C are C-connected or not.

1/ There are two vertices x and x' of C which are not C-connected. Figure 6.8 serves as an illustration of the following reasoning.

Denote by y and z the two neighbours of x on C. Trivially, x, y, z are all C-connected to x. As x is not C-connected to x', we can consider P' the longest path portion of C containing x' whose inner vertices are all not C-connected to x. Let P be the path of C such that $C = PP'$.

We prove that if u and v are inner respectively in P and P', then they are not C-connected.

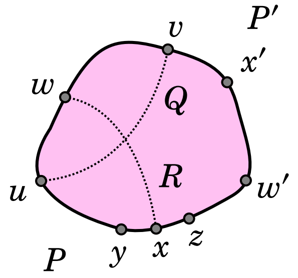


Figure 6.8: The outer cycle is C, and P and P' are the two distinct w-w' paths on C.

If $u = x$, by definition, inner vertices of P' are not C-connected to x.

Next, consider $u \neq x$. For the sake of contradiction, let Q be a C-path from u to v. Let w be the endpoint of P such that u is inner in wPx, and let w' be the other end of P. By minimality of P, there exists an x-w path R inner disjoint from C. If Q and R were not disjoint, then they would induce an x-v path inner disjoint from C, a contradiction. Otherwise, (Q, R) is a crossing of C in G, a contradiction.

We have just shown that $\{w, w'\}$ separates C in G in at least two components, respectively containing P and P'. In particular the edge ww' cannot be used to construct a crossing of C in G. By maximality of G, $ww' \in E(G)$, and we can see G as given by the edge composition of two graphs along ww' respectively containing P and P'. Quite directly, these two graphs are maximally crossless for Pww' and $P'ww'$, respectively. By induction hypothesis they are webs and, by Proposition 6.7.1, G is a web.

2/ All pairs of distinct vertices of C are C-connected.
Take consecutive vertices x, y, z on C with an x-z path P inner disjoint from C. For such a path P we write C_C^P for the component of $C - x - y - z$ in $G - P$. We assume that P is minimal with regard to the measure

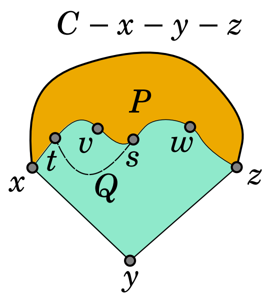
$$\mu(P) = (|G| - |C_C^P|, |P|)$$

where the order on $\mathbb{N} \times \mathbb{N}$ is the lexicographic product of the usual order $<$. Said otherwise we choose P to maximise the order of the component of $G - P$ containing $C - x - y - z$ while minimising the length of P. In particular, P is induced.

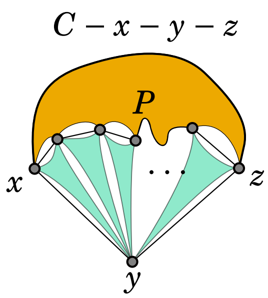
Take u an inner vertex of P. We prove that u is $C \cup P$-connected to $C - x - y - z$. For the sake of contradiction, assume not. Let $v, w \in P$ be such that vPw contains u as an inner vertex and is the longest subpath of P whose inner vertices are all not $C \cup P$-connected to $C - x - y - z$. As P is induced and as vPw contains at least u as an inner vertex, G does not contain vw.

Consider $G+vw$. By maximality of G, C is crossed in $G+vw$. Let (P_1, P_2) be a crossing of C in $G+vw$, with, say, $vw \in E(P_1)$. The path P_2 must contain some inner vertex of vPw as otherwise (P_2, P_1vPwP_1) would be a crossing of C in G.

One end of P_2 at least must be on $C-y$. Without loss of generality, assume this is the case of its last vertex. Let s be the last vertex in vPw seen by P_2. As the last vertex of P_2 is on $C-y$ and as s is not $C \cup P$-connected to $(C-x-y-z)$, sP_2 must either intersect $C \cup P$ at one of its inner vertex t or its last vertex is x or z. In the first case, as P_2 is a C-path, we actually have $t \in V(P)$. In any case, there is a vertex t of sP_2 appearing after s that is on P. By maximality of s it must be outside of vPw, providing an s-t path Q such that either $v \in sPt$ or $w \in sPt$. But then, P can be replaced by $PsQtP$ giving a path with lower measure μ as C_C^{PsQtP} contains v or w on top of all vertices of C_C^P. This is a contradiction.



The fact that all inner vertices of P are $C \cup P$-connected to $C-x-y-z$ has the following implication: components of $G-P-y$ that are incident to y are incident to at most two vertices of P and those are consecutive in P. Indeed if such a component could join two non-consecutive vertices s and t of P in a $C \cup P$-path Q not intersecting C_C^P, then $PsQtP$ would give a better path than P w.r.t the measure μ. Said otherwise, we have proved that G has the following shape:



i.e. it is the path composition along P of a spanning subgraph of a web (made of the green triangular shapes; its rib is a fan graph) with frame $zyxP$ with a graph H (the orange area above the triangular shapes) containing $PzCx$ as a cycle. It is easily proved that in any edge extension of H, $PzCx$ is crossed. Hence H is a web with frame $PzCx$, and by Proposition 6.7.1, since G is the path composition along and induced path, of H and a spanning subgraph of a web, G is a spanning subgraph of a web with frame C. By maximality of G and Lemma 6.11.1, G is a web with frame C. □

Combining Lemmas 6.11.1 and 6.12.1, with Proposition 6.8.1 and Theorem 6.4.1, the generalised two paths theorem, is proved.

In the next chapter we use the two paths theorem to explore the hard graphs of treewidth at most four.

SOLVING THE TWO DISJOINT PATHS PROBLEM

6.13 The two disjoint paths problem. We summarise the arguments for the proof of Lemma 6.12.1. Given a graph in which a cycle C is crossless such that C is crossed in any edge extension of the graph, we consider an induced C-path P between non-consecutive vertices of C. If P does not separate C then C is crossed in G. Otherwise, in $G - P$, C is divided into two paths P_1 and P_2 corresponding to two components V_1 and V_2 of $G - P$; up to some assumptions on P, the graphs $G[V_1 \cup P]$ and $G[V_2 \cup P]$ are subinstances on which we apply the induction hypothesis.

We now turn this inductive argument into a recursive algorithm for finding crossings. The minimality assumption on separating paths (with respect to the measure μ) in the proof of Lemma 6.12.1 is not suitable to get an efficient algorithm; the difficulty is to find other separating paths with "good enough properties".

Let G be a graph with a tuple T of distinct vertices. A *web completion* is a set F of edges such that $G + F$ is a web with frame T.

By Theorem 6.4.1 web completions exist if and only if T is crossless; thus we generalise and strengthen the two disjoint paths problem as follows:

Two Disjoint Paths or Web Completion
Input: A graph with a tuple of distinct vertices.
Output: A crossing or a web completion.

In the next two sections we only consider the decision version of this problem. We discuss the actual computation of crossings and web completions in §6.16, and algorithmic complexity in §6.17.

6.14 Method. Henceforth we assume a tuple $T = (x_1, \ldots, x_k)$ of distinct vertices in a graph G. A set of edges F is called *safe* for T when T is crossed in $G + F$ implies that T is crossed in G.

If T is crossed then all sets of edges are safe. As observed in the proof of Lemma 6.12.1, the edges $x_i x_{i+1}$ ($1 \leq i \leq k-1$) and $x_k x_1$ are all safe for T. Hence we assume that T induces a cycle C in G.

The goal is to devise a divide-and-conquer strategy whose recursive case can be outlined as follows: given a graph G with a cycle C, find two graphs G_1 and G_2 with cycles C_1 and C_2 such that:

- G is the path composition of G_1 with G_2, and
- C is crossed in G if and only if C_1 or C_2 is crossed, respectively in G_1 and G_2.

There are two base cases: 1/ if all C-paths of G are between consecutive vertices of C then C is crossless; and 2/ given a C-path P between non-consecutive vertices of C, if $C - P$ is connected in $G - P$, then C is crossed in G.

Proposition 6.14.1. *Let P be a C-path of G between non-consecutive vertices of C, and P_1 and P_2 the two distinct subpaths of C sharing their endpoints with P and such that $C = P_1\overline{P_2}$.*

If P separates $\mathring{P}_1$ from $\mathring{P}_2$ then P_1 and P_2 are contained in subgraphs G_1 and G_2 of G, respectively, and G is the path composition of G_1 and G_2 along P.

Proof. Observe that $\mathring{P}_1$ and $\mathring{P}_2$ are both non-empty as the endpoints of P are non-consecutive on C.

The statement follows by studying components of $G - P$. Any such component containing vertices of $\mathring{P}_i$ is associated to G_i. All other components as well as edges between non-consecutive vertices of P are arbitrarily distributed between G_1 and G_2. The graph G_i is defined as the subgraph of G induced by the union of P with the components of $G - P$ associated with G_i to which any edge between non-consecutive vertices of P it is not associated with is removed. The fact that G is indeed the path composition of G_1 and G_2 along P follows easily from P being a separator of $\mathring{P}_1$ and $\mathring{P}_2$. □

Following the divide-and-conquer strategy described above, we regard as subinstance the pairs (G_i, C_i) (with $C_i = P_i P$). Unfortunately, in general it is false that C_1 or C_2 is crossed (resp. in G_1 and G_2) if and only if C is crossed in G.

The left-hand side example of Figure 6.9 shows a graph G given as the path composition of two graphs G_1 and G_2, each corresponding to a coloured area of the drawing (two edges do cross so this is indeed a drawing and not a plane embedding). The path P is the one in the intersection of the two graphs. The cycle C is the outer 4-cycle. The cycle C_1 is crossed in G_1 while C

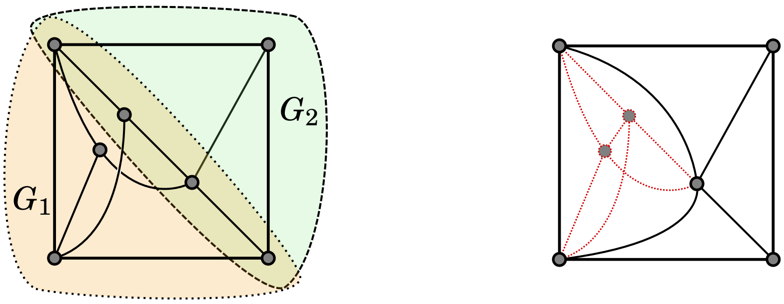


Figure 6.9: A graph G and its only web completion with respect to its outer 4-cycle.

is crossless in G. On the right-hand side of Figure 6.9 we have displayed the only web completion of G (see how it can be built by adding one clique of order two on some triangle of the only rib of order five with a frame of length four in Figure 6.1). It cannot be expressed as the path composition of two graphs along P: somehow the wanted subinstances induced by P in G do not behave well with regard to web completions.

Our solution is ensuring a connectivity property on P.

Given $i \in \{1,2\}$, we say that a vertex $y \in P$ *is on P-side i* when y and $\mathring{P}_i$ are P-connected.

Proposition 6.14.2. *Let P be a C-path of G between non-consecutive vertices of C, and P_1 and P_2 the two distinct subpaths of C sharing their endpoints with P and such that $C = P_1\overline{P_2}$.*

If P separates $\mathring{P}_1$ from $\mathring{P}_2$ and all vertices of P are on both P-sides, then C is crossed in G if and only if P_1P is crossed in G_1 or P_2P in G_2.

Proof. Suppose that P_1 and P_2 are separated by P, and that all vertices of P are on both P-sides. We prove the contrapositives of both implications.

- Assume that P_iP is crossless in G_i for all $i \in \{1,2\}$. By Theorem 6.4.1, G_i has a web completion F_i.

  In $G_i + F_i$, the path P must be induced. Indeed, if xy were an edge of $G_i + F_i$ between non-consecutive vertices of P, then we could consider z an inner vertex of xPy. As z is on both P-sides, there is in G_i a $\mathring{P}_i$-z path Q. The pair (Q, xy) is a crossing of P_iP in the web $G_i + F_i$, a contradiction.

  By Proposition 6.7.1 the web composition of $G_1 + F_1$ with $G_2 + F_2$ along P is a web, proving that G has a web completion in the form of $F_1 \cup F_2$, and as a consequence of Theorem 6.4.1 that C is crossless.

- Suppose C is crossless.

  Consider a component D of $G-P$ containing vertices of neither P_1 nor P_2. It is incident to at most two vertices of P and those must be consecutive. Indeed, if that was not the case then D would induce an x-y path Q between non-consecutive vertices of P. Let z be an inner vertex of xPy. As z is on both P-sides, there are $\mathring{P}_1$ -z ad z-$\mathring{P}_2$ paths R and S inner disjoint with C, P, and Q. The pair $(PxQyP, RS)$ is a crossing of C in G, a contradiction.

  The same method works if D is replaced with an edge between non-consecutive vertices of P.

  In particular, P is induced in G.

  Let $\mathcal{D}$ be the set of all components of $G-P$ containing vertices of neither P_1 nor P_2. C is crossless in $G - \mathcal{D}$, so by Theorem 6.4.1 we let F be an associated web completion. As edges between non-consecutive vertices of P make C be crossed, P remains induced in $(G - \mathcal{D}) + F$. Hence P's vertices are all in the rib of $(G-\mathcal{D})+F$. This gives webs H_i with frames P_iP such that $(G - \mathcal{D}) + F$ is their web composition along P. Obviously H_i is a web containing $G_i - \mathcal{D}$. For each component D of $\mathcal{D}$ incident to, say, an edge xy of P, if its vertices are in G_i then we make them all into a clique, and adjacent to all vertices of the triangle of H_i's rib containing xy as well as to all vertices of the clique of this triangle.

  This construction provides a web completion of both G_1 and G_2 proving that P_1 and P_2 are crossless, respectively in G_1P and G_2P. □

The example of Figure 6.9 proves that for some graphs no C-path P may be used to apply this proposition and get two subinstances.

Given a C-path P of G we call *P-completion* a safe set of edges F for C such that $P+F$ contains a C-path P' whose vertices are all on both P'-sides in $G+F$.

Proposition 6.14.3. *Let G be a graph with a cycle C. For any C-path P between non-consecutive vertices of C there exists a P-completion.*

Proof. Assume $P = y_1 \dots y_l$. If C is crossed in G then y_1y_l is a P-completion. If instead C is crossless, then by Theorem 6.4.1 there is a set F of edges such that $G+F$ is a web. Any C-path of $G+F$ which is induced and contained in $P+F$ can then be used to prove that F is a P-completion (see Claim 6.11.2 for details on proving that vertices of P' lie on both P'-sides). □

We now introduce our general method in Algorithm 1. We assume that a call to **P_completion**(G, C, P) computes a P-completion F, the associated

C-path P', and returns the pair $(G + F, P')$. Up-to some implementation of **P_completion**, by Proposition 6.14.2 and induction on the size of the recursive instances, the method is both correct and terminating. A brutal computation of P-completions amounts to solving directly the two disjoint paths problem. This algorithm is only of interest if P-completions can be computed efficiently. This is the topic of the next section.

Algorithm 1 General method for the two disjoint paths decision problem.

procedure 2DP(G, C) ▹ decides if C is crossed in G or not
    if no C-paths between non-consecutive vertices of C exist **then**
        return False
    else
        $P \leftarrow$ a C-path between non-consecutive vertices of C
        if P does not separate C in G **then**
            return True
        else
            $G, P \leftarrow$ **P_completion**(G, C, P)
            $G_i, C_i \leftarrow$ two subinstances induced by G, P and C
            return (2DP(G_1,C_1**)** $\vee$ **2DP(**G_2,C_2**))**

6.15 Computing P-completions efficiently. Recall that we are working with a graph G with a cycle C and a C-path P between non-consecutive vertices of C. The goal is to compute a P-completion efficiently. As before we let P_1, P_2 be the two distinct subpaths of C with the same endpoints as P and such that $C = P_1\overline{P_2}$.

The core idea behind computing a P-completion is to determine which P-sides each vertex of P lies on. This reveals some gaps as in the left-hand side graph shape of Figure 6.10. The outer cycle is C, the central path with explicit vertices is P. Roughly, the corresponding red dotted edges form a P-completion. The issue is that the red dotted edges may not be well-defined, because there might be components of $G - P$ intersecting neither $\mathring{P}_1$ nor $\mathring{P}_2$, and also because the gaps may not intersect on vertices which are on both P-sides. The right-hand side graphs of Figure 6.10 depicts two possible choices for the red dotted vertices of a graph (where P is the dashed path and C is the outer cycle). In the top part one red dotted edge is unsafe, and in the bottom part, the two red dotted edges cannot both be used in a C-path. The solution, described in Lemma 6.15.1, is to add red edges one after the other, computing the next red edge at each step.

Write $y_1 \ldots y_l$ for P. Since P connects non-consecutive vertices of C, both $\mathring{P}_1$ and $\mathring{P}_2$ are non-empty. As y_1 (resp. y_l) is adjacent to a vertex of both $\mathring{P}_1$

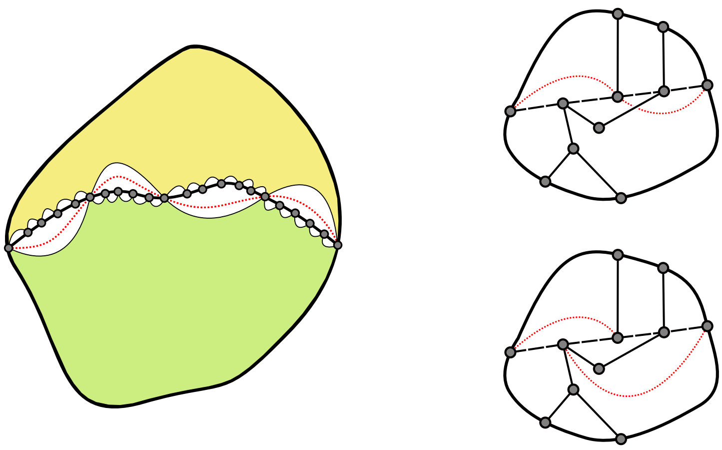


Figure 6.10: Intuition on how P-completions are computed

and $\mathring{P}_2$, y_1 (resp. y_l) is on both P-sides. Hence the following are well-defined: 1. the largest prefix Py of P whose vertices are all on both P-sides; 2. the first vertex z on P found after y which is on a given P-side.

Lemma 6.15.1. *Assume that not all vertices of P are on both P-sides. Let the following be defined:*

- *j with $1 \leq j < l$ the greatest index such that all vertices of Py_j are on both P-sides,*
- *r with $j < r \leq l$ the lowest index such that y_r is on at least one P-side, and,*
- *s with $r \leq s \leq l$ either:*
  - *$s = r$ if r is on both P-sides, or*
  - *the lowest index such that y_s lies on the P-side r does not lie on.*

Then the edge y_jy_s is safe, and in the graph $G + y_jy_s$, all vertices of Py_jy_s lie on both Py_jy_sP-sides.

Proof. To prove that all vertices of Py_jy_s are on both Py_jy_sP-sides in $G+y_jy_s$ we only need proving that y_s is on both Py_jy_sP-sides. The remainder follows from the assumption that all vertices of Py_j lie on both P-sides in G.

Without loss of generality, we assume that y_r is on P-side 1 in G. The vertex y_s is on P-side 2 (we may have $r = s$). These two facts provide a $\mathring{P}_1$-y_r path Q inner disjoint with $P \cup C$ and a $\mathring{P}_2$-y_s path R inner disjoint with $P \cup C$. The paths R and Qy_rPy_s prove that y_s is on both Py_jy_sP-sides in $G + y_jy_s$.

We now prove that $y_j y_s$ is safe.

If C is crossed in G then it remains crossed in $G + y_j y_s$.

Otherwise, C is crossless in G and we prove that it is crossless in $G + y_j y_s$ too.

By Theorem 6.4.1, G has a web completion F. Below, we modify F to get a P-completion that contains $y_j y_s$, proving this edge is safe for C. In $G + F$, we consider an induced C-path P' extracted from $P + F$. We first prove four claims relating P' with P:

Claim 6.15.2. The edge $y_j y_r$ is safe in G.

Claim proof. For the sake of contradiction, let (Q, R) be a crossing of C in $G + y_j y_r$.

Since C is crossless in G, (Q, R) is not a crossing of C in G and (without loss of generality) Q contains the edge $y_j y_r$. As vertices y_t with $j < t < r$ have no P-sides, $\{y_j, y_r\}$ separates them from C. Since R has its endpoints on C, R does not intersect $y_j P y_r$. The pair $(Q y_j P y_r Q, R)$ is a crossing of C in G, a contradiction. ◁

Claim 6.15.3. If some vertex z has P-side i in G then it has P'-side i in $G + F$.

Claim proof. Indeed, as $V(P') \subseteq V(P)$, a $C \cup P$-path from z to $\mathring{P}_i$ in G is a $C \cup P'$-path from z to $\mathring{P}_i$ in $G + F$. ◁

Claim 6.15.4. All vertices y_t with $1 \leq t \leq j$ are on P'. In particular, the prefix of P' of length $j - 1$ is $P y_j$.

Claim proof. For the sake of contradiction, let y_t with $1 \leq t \leq j$ be a vertex which is not on P'. Since y_t is on both P-sides, let Q_1 and Q_2 be $P \cup C$-paths in G, respectively from $\mathring{P}_1$ to y_t and from y_t to $\mathring{P}_2$. The pair $(Q_1 Q_2, P')$ is a crossing of C in $G + F$, a contradiction.

We prove that the prefix of P' of length $j - 1$ is $P y_j$.

For the sake of contradiction, assume that the prefix of P' of length $j - 1$ is not $P y_j$ and let y_a be the first vertex on P' such that $P' = P y_a y_{a'} P'$ with $a' \neq a + 1$. As y_{a+1} is on both P-sides, we can consider $P \cup C$-paths Q and R respectively from $\mathring{P}_1$ to y_{a+1} and from y_{a+1} to $\mathring{P}_2$. The pair $(Q_1 Q_2, P y_a y_{a'} P)$ is a crossing of C in $G + F$, a contradiction. ◁

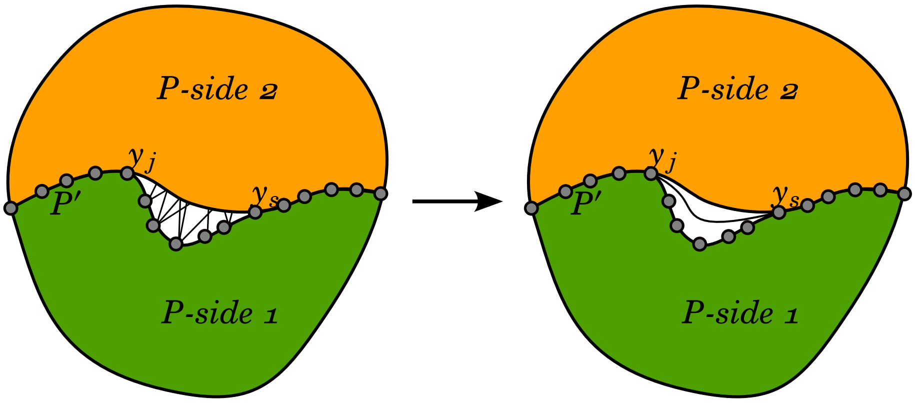
Claim 6.15.5. The $j + 1$th vertex of P' is a vertex y_t with $j < t \leq s$.

Claim proof. For the sake of contradiction, assume not. By Claim 6.15.4, the $j + 1$th vertex of P' is y_t for $t > s$. Since y_r and y_s are respectively on, say, P-sides 1 and 2, we can consider $P \cup C$-paths Q and R respectively from $\mathring{P}_1$ to y_r and from $\mathring{P}_2$ to y_s. The pair $(Q y_r P y_s R, P y_j y_t P)$ is then a crossing of C in $G + F$, a contradiction. ◁

We are now back to proving that $y_j y_s$ is safe. By Claim 6.15.2 we can assume $r \neq s$, implying, by definition of r and s, that r is on exactly one P-side in G. Without loss of generality, assume y_r is on P-side 1 but not 2 in G. This implies that y_s is on P-side 2 in G and is the only such vertex among $y_{j+1}, \ldots, y_s$. Claims 6.15.4 and 6.15.5 prove that $P' = P y_j y_t y_{t'} P'$ with $j < t \leq s$.

We prove the safeness of $y_j y_s$ by induction on the length of P'. If $s = t$ then obviously F contains $y_j y_s$ and this edge is safe. Assume $s - t > 0$. It implies that y_t is not on P-side 2 in G.

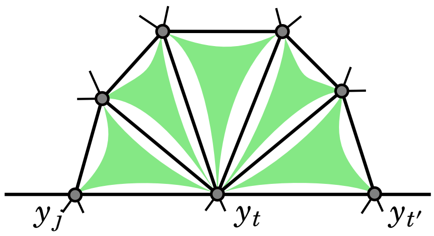
The idea is using the fact that none of the vertices in $y_{j+1}, \ldots, y_t$ lie on P-side 2 in G to "cut" some edges of F between these vertices and P-side 2, and replace them by an edge closer to $y_j y_s$:



Since this illustration takes the planar embedding of the rib of G into account, adding $y_j y_s$ is safe after the removal of the other edges.

This illustration works when $P' = P$ and the vertices in $y_{j+1}, \ldots, y_{s-1}$ are directly connected to vertices on P-side 2 (and hence the edges we remove are all incident to vertices in $y_{j+1}, \ldots, y_{s-1}$). Neither one of these assumptions hold in general. Instead we prove that $y_j z$ is safe, for z a vertex appearing after y_t on P', allowing us to apply the induction hypothesis on $P' y_j z P'$.

By Proposition 6.3.3, in $G + F$, y_t and neighbours of y_t that are on P'-side 2 and on the rib of $G + F$ describe a fan graph:

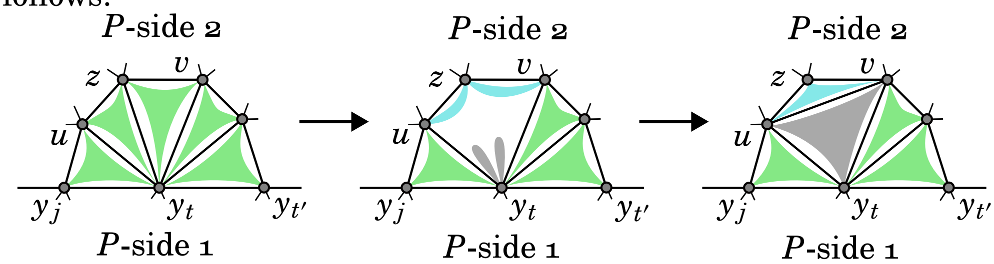


In this figure, the green triangles correspond to the clique vertices of the web $G+F$. The path of this fan graph is a y_j-$y_{t'}$ path Q. In what follows we assume we have a plane drawing of $G+F$ which induces a plane embedding of its rib, and such that clique vertices and associated edges of a given triangle of the rib are restricted to the area of the plane bounded by the said triangle.

Let z be an inner vertex of Q on P-side 2 in G. We prove $zy_t \in F \setminus E(G)$.

Write u and v for the predecessor and successor of z on Q, respectively. The edge $y_t z$ must be in $F \setminus E(G)$ (zy_t is an edge we want to remove, as explained above). Together with a P-path from z to $\mathring{P}_2$, having $zy_t \in E(G)$ proves that y_t is on P-side 2, which is a contradiction in G.

We describe a process modifying F to remove z from y_t's neighbours. Remove $y_t z$ from F as well as the edges between clique vertices of the triangles uzy_t and $zy_t v$ of $G+F$'s rib linking a vertex on no P-side and one on P-side 2 in G. Add the edge uv and fill back the cliques in a compatible way with respect to the new triangles uzv and $uy_t v$. The whole process is represented as follows:



In this figure, the grey loops are the clique vertices on no P-sides in G, and the blue thick arcs the ones that have P-side 2 in G.

The graph obtained at the end of this process is a web. Its rib is induced by the same vertices as the rib of $G+F$. The only possible issue would be that uv is already an edge in $G+F$. That cannot be as then in the drawing of $G+F$ fixed above, uvy_t would have been a triangle of the rib while not being an inner face.

We distinguish several cases depending on which P-sides in G the inner vertices of Q are on. In the first case, we are able to remove edges in a way close to the illustration provided above, and hence we obtain that $y_j y'_t$ is safe. In the second case, more edges needs to be removed to prove that $y_j y_{t'}$ is safe.

- **All the inner vertices of Q are on P-side 2 in G;** after applying the process above $|Q|-2$ times, we get a web containing G as well as the edge $y_j y_{t'}$, proving its safeness. In this web the path $P' y_j y_{t'} P'$ is smaller in length than P' allowing us to conclude by induction hypothesis.

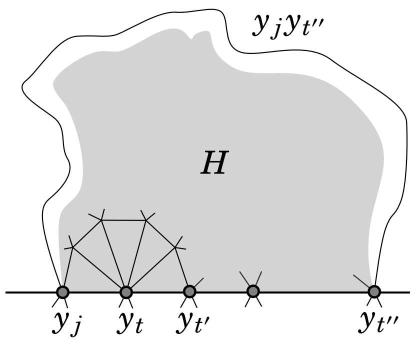
- **There exists an inner vertex of Q on no P-side in G;** by applying the process above repeatedly, we can assume that all inner vertices of Q have no P-side in G.

  Consider H the subgraph of $G+F$ induced by the vertices that are $C \cup P'$-connected to y_t, on P'-side 2 in $G+F$, as well as on no P-side in G. All inner vertices of Q are in H. Hence, H is connected.

  The first vertex of P' that is adjacent to a vertex of H is y_j. Indeed, if it was a vertex y_a with $a < j$, then we could consider P-paths Q_1 and Q_2 in G from $\mathring{P}_1$ to y_j and from y_j to $\mathring{P}_2$ respectively and R a y_a-y_t path in H, and (Q_1Q_2, Py_aRy_tP) would be a crossing of C in $G+F$, a contradiction.

  Let $y_{t''}$ be the last vertex of P' that is adjacent to some vertex of H.

  We prove that $y_jy_{t''}$ is safe by modifying F in order to add $y_jy_{t''}$. In $G+F$, we remove all edges between vertices of H and vertices of $G+F$ that are not on P'. Those are edges between vertices having no P-side and on P-side 2 in G: they are all in $F \setminus E(G)$. Next we add the edge $y_jy_{t''}$. In the plane drawing of $G+F$ we do that close enough to H making it clear that the cycle $y_jP'y_{t''}y_j$ separates vertices of H from the rest of the graph:



  Let G' be the obtained graph. It is clear that G is a spanning subgraph of G'. We prove that C is crossless in G'. By contradiction, assume that (R, S) is a crossing of C in G'. One of the paths, say R, must contain $y_jy_{t''}$. The path S must contain some vertices of $y_jP'y_{t''}$ as otherwise $(Ry_jP'y_{t''}R, S)$ would be a crossing of C in $G+F$. Let y_a and y_b be the first and last inner vertices of $y_jP'y_{t''}$ appearing on S. As the cycle $y_jP'y_{t''}y_j$ separates H from the rest of G', and from C in particular, and as the endpoints of S are on C, the path $Sy_aP'y_bS$ intersect H. As H is connected we consider a y_j-$y_{t''}$ path Q' such that $\mathring{Q}'$ is a path in H. The pair $(Ry_jQ'y_{t''}R, Sy_aP'y_bS)$ is a crossing of C in $G+F$, a contradiction.

Take a web completion of G' and replace the path P' by $P'y_jy_{t''}P'$. The latter is smaller in length than P', allowing us to conclude by induction hypothesis. □

We can compute P-completions inductively using Algorithm 2. Its termination is ensured by the fact that, in the while loop, the length of P is non-increasing while the longest prefix Py of P whose vertices are all on both P-sides sees its length increase at each step, as a consequence of Lemma 6.15.1. Its correctness is a direct consequence of this lemma.

Algorithm 2 Computing P-completions.

procedure P_COMPLETION(G, C, P)
    $P_1, P_2 \leftarrow$ the two subpaths of C separated by P with $C = P_1\overline{P_2}$
    $y \leftarrow$ first vertex of P
    while y is not the last vertex of P **do**
        $z \leftarrow$ first vertex of P strictly after y that is on some P-side i
        $u \leftarrow$ first vertex of P after z on P-side $j \neq i$
        $G \leftarrow G + yu$
        $P \leftarrow PyuP$
        $y \leftarrow$ first vertex of P after y whose successor is not on both P-sides
    return G, P

6.16 Extracting crossings and web completions. In the previous two sections we have shown how to decide the 2-DISJOINT PATHS AND WEB COMPLETIONS problem. Now, we show how Algorithm 1 can be refined to compute crossings or web completions explicitly. We use the same notations G, C, P, P_i, G_i, C_i as above.

When Algorithm 1 returns True we compute a crossing of C as follows:

- if the C-path P chosen in G does not separate C then we compute a $\mathring{P}_1$-$\mathring{P}_2$ path P' in $G - P$ and return (P, P').
- if some recursive call **2DP**(G_i, C_i) returns True, and hence returns a crossing (Q, R) of C_i in G_i, its shape must fall into one of the five cases of Figure 6.7. The solutions presented in this figure to build a crossing of C in G still work here thanks to vertices of P being all on both P-sides. Returning this crossing then reduces to computing appropriate $C \cup P$-paths from Q or R to $\mathring{P}_1$ and/or $\mathring{P}_2$.

In the case Algorithm 1 returns False, we compute web completions as follows:

- if there are no C-paths in G between non-consecutive vertices of C, then each component of $G - C$ is incident to either one or two vertices of C, and in the latter case the two vertices must be consecutive in C. We choose $x_0 \in V(C)$ and let be $F = \{x_0 y \mid y \in V(C),\ y \neq x_0\} \cup C$. The graph $R = (V(C), F)$ is a rib compatible with G. For each component of $G - C$ we choose a triangle of R it is incident to. For each triangle T of R we turn the components of $G - C$ we associated with T into a clique on T. This provides a web completion of G.

- if both calls **2DP**(G_i, C_i) returned False, and hence a web completion of the graph G_i with respect to C_i for $i \in \{1, 2\}$, then their union is a web completion of G with respect to C.

6.17 Complexity. Let G be a graph with a cycle C and a C-path P between non-consecutive vertices of C. To analyse the complexity of the whole algorithm, we need to be more precise about:

- how do we compute paths,
- how do we decide whether a path separates C, and
- how do we decide P-connectedness and in particular which P-sides a given vertex belongs to.

All of these can be done using a *search algorithm* (e.g. depth- or breadth-first searches) forbidding the traversal of some vertices $X \subseteq V(G)$. We call *X-searches* such searches.

Whereas a usual search algorithm allows one to compute the connected components of a graph, X-searches allow for the computation of components of $G - X$ as well as the subsets of X these components are incident to.

In other words, X-searches allow us to compute the prime components of (G, X).

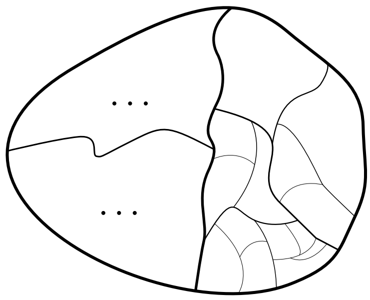
Let $n = |V(G)|$ and $m = |E(G)|$.

Proposition 6.17.1. *There exists an implementation of Algorithm 1 that runs in $O(nm)$ time on connected graphs.*

Proof. Let (G, C) be an instance of **2DP** such that G is connected. An X-search can be done in $O(m)$ assuming that the relation $- \in X$ is an attribute of vertices of G accessible in constant time. Indeed, this is a simple variation of usual search algorithms.

Beside the call to **P_completion**(G, C, P) the complexity of one recursive call to Algorithm 1 lies in four lines: the two **if** statements, the first computation of P, and the computation of the two subinstances. Each can be done in $O(m)$ time by using either a C-search or a P-search.

Hence, if p is the number of recursive calls made to **2DP** and K the complexity of all calls to **P_completion**, Algorithm 1 runs in $O(pm+K)$. Each time a new call to **2DP** is made, it means an instance has been divided in two subinstances by finding a C-paths P separating C. It is not difficult to see that the union C of all cycles passed as arguments of **2DP** is a planar graph: following the recursive calls, we start with a cycle and we add paths separating inner faces recursively:



In the figure, the outer cycle is the first cycle C, while inner C-paths are the recursive separating paths P computed in the different calls of **2DP**. The thinner the line the deeper the recursive call.

The number p of calls to **2DP** is bounded by the number of edges of this planar graph, meaning $p = O(n)$. Hence the complexity of Algorithm 1 is $O(nm + K)$.

Finally we look at Algorithm 2. It requires one P-search at the beginning to compute P_1 and P_2, as well as one $(C \cup P)$-search for each step of the while loop to compute the P-sides each vertex is on. The number of such searches among all calls to **P_completion** in a run of **2DP** is bounded by the sum of p with the number of edges added by **P_completion**. Both are bounded by the number of edges in C which is a $O(n)$. Hence, $K = O(nm)$ and the complexity of Algorithm 1 is $O(nm)$. □

To compute crossings or web completions as described in §6.16, the overall complexity is not modified. We have to be careful about how to represent web completions, however.

Indeed, imagine that *most* vertices of G are in a component D of $G-C$ that is incident to exactly one vertex of C ($n = |G| = O(|D|)$). If $G-D$ is a rib, then in any web completion of G, all vertices of D are in the clique of some triangle of $G-D$. This clique contains $\Omega(|D|^2) = \Omega(n^2)$ edges. If D is sparse, say it is a tree, then G takes only $O(n)$ space in memory to be stored using adjacency lists while all its web completions have $\Omega(n^2)$ edges.

To avoid this problem, instead of representing a web using adjacency lists, we distinguish between rib and clique vertices. The adjacencies between rib vertices are kept in the form of adjacency lists, while the adjacencies of clique vertices are held in the form of the three rib vertices forming their corresponding triangle. As ribs are planar they have $O(n)$ edges. This yields for a $O(n)$-space representation of webs in memory.

7
Towards an axiomatisation at treewidth at most four

7.1 Introduction. Proposition 5.7.3 is a key lemma in our analysis of hard graphs of treewidth at most three. It describes the only two shapes of graphs of arity three without K_3 as a sourced minor. To solve the treewidth axiomatisation conjecture for graphs of treewidth at most four, we aim to analyse the hard graphs of treewidth at most four. A natural direction is to seek a characterisation to Proposition 5.7.3, but in the case of graphs of arity four (excluding K_4 as a sourced minor instead of K_3).

Such a characterisation is provided by Fabila-Monroy and Wood in [29]. Its proof relies heavily on the two path theorem.

In the second section, we refine the result of Fabila-Monroy and Wood for graphs of treewidth at most four (§7.4 to §7.6). We begin, in the first section, by lifting the two paths theorem to hypergraphs (§7.2 and §7.3).

In the third and final section, we use our refinement of the result of Fabila-Monroy and Wood to sketch the beginning of an analysis of hard graphs of treewidth at most four (§7.7 and §7.8) in which the definition of XG_k and its use in the proof of incompleteness of C find their origin.

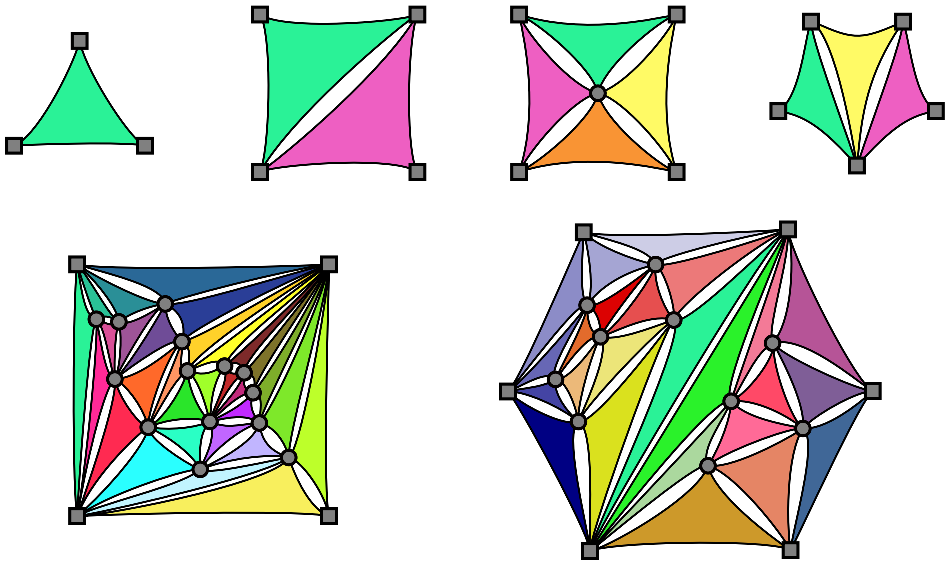


Figure 7.1: Hyperwebs corresponding to the webs in Figure 6.1

We conclude the thesis in §7.9.

TWO PATHS THEOREM FOR HYPERGRAPHS

The two paths theorem as presented in the previous chapter states that webs are precisely the maximally (edge-wise) crossless graphs (with regard to their frame). Hypergraphs allow parallel hyperedges. Hence, the notion of "edge maximal" graphs is not defined for hypergraphs. Instead, to prove a two paths theorem for hypergraphs, we introduce *hyperwebs*, a generalisation of webs to hypergraphs, and use them as canonical shapes, replacing the maximality condition.

In this chapter, by "graph", we mean a hypergraph as introduced in Definition 3.2.1.

7.2 Sourced webs. We call *sourced rib* (resp. *sourced web*) a rib (resp. a web) in which the frame vertices are promoted to sources, the interface order respecting that of the frame as a cycle.

We call *hyperweb* a graph H obtained from a sourced rib R by replacing the triangles of R with edges of arity three.

Examples of hyperwebs are depicted in Figure 7.1.

Proposition 7.2.1. *A graph has a hyperweb shape if and only if its footprint is a spanning subgraph of a sourced web.*

Proof. Let G be a graph.

Suppose that G has the shape of a hyperweb W and let σ be a graph substitution such that $G \simeq \sigma(H)$. Let W' be the sourced web defined by filling each triangle T of the footprint of W (which is a rib by definition) corresponding to the edge a of W with a clique on $|\sigma(a)|$ vertices. The footprint of G is a spanning subgraph of W'.

Suppose that the footprint of G is a spanning subgraph of a sourced web W. Let W' be the hyperweb associated to W's rib. Then G has a W'-shape via the substitution mapping a variable a label of an edge of W' corresponding to a triangle T of W to the subgraph of G corresponding to the clique of W filling T. □

7.3 Two paths theorem with shapes. Given a graph G of arity n with interface $(s_1, \dots, s_k)$, we say that G is *crossed* when there exists two vertex-disjoint paths in G, one from s_i to s_j and the other from s_r to s_s, with $i < r < j < s$. We refer to the corresponding pair of paths as a *crossing*.

Observe that the paths of a crossing may use common edges, though via distinct pairs of incident vertices. For example, in the atomic graph of arity four with interface (s_1, s_2, s_3, s_4), the pair of paths (s_1s_3, s_2s_4) is a crossing, and both paths use the unique edge of the graph.

Theorem 7.3.1 (Two paths theorem for hypergraphs).
A graph is either crossed or has a hyperweb shape, one statement excluding the other.

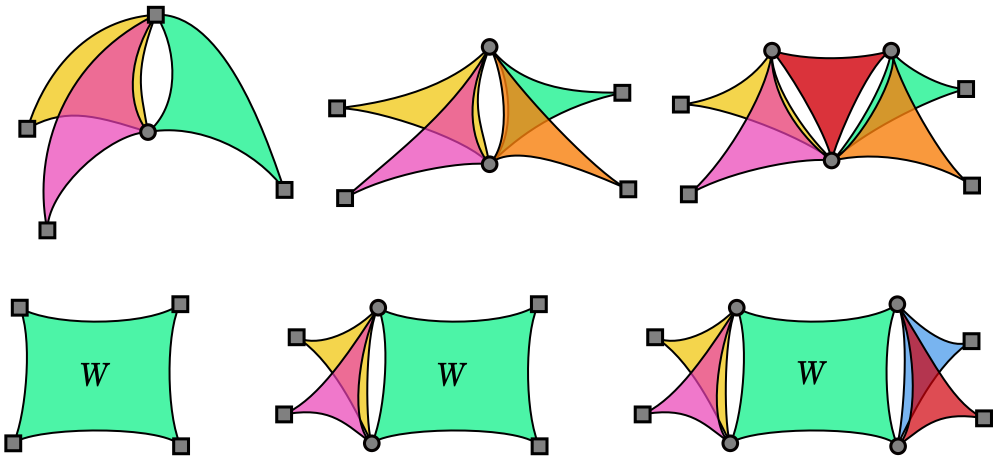
Moreover, if u and v are non-adjacent vertices in a hyperweb W, then $W+uv$ is crossed.

Proof. Direct consequences of Proposition 7.2.1, Theorem 6.4.1, and of the fact that two vertices of a graph are connected by a paths if and only if they are connected in the footprint of the graph. □

SOURCED MINORS AND K_4

7.4 K_4 sourced minors and shapes.

Theorem 7.4.1. *Every graph of arity four has either K_4 as a sourced minor, or one of the following six shapes:*



where the 4-edges labelled W are 4-hyperwebs.

Moreover, none of these shapes has K_4 as a sourced minor.

Proof. The statement is a translation of [29, Theorem 15] obtained via a (direct) generalisation of Proposition 7.2.1 for the obstructions appearing in [29, Theorem 15]. □

To obtain the completeness of F_3, in the proof of Theorem 5.2.1, we used the characterisation of graphs of arity three without K_3 as a sourced minor (Proposition 5.7.3) to refine the shapes combining separation pairs in hard graphs of treewidth at most three (Lemma 5.5.1).

An important property of the two shapes of graphs of arity three without K_3 as sourced minor given in Proposition 5.7.3 is that they have treewidth at most three.

Theorem 7.4.1, which characterises graphs of arity four without K_4 as a sourced minor via some shapes, can be seen as a generalisation of Proposition 5.7.3. A limitation of the theorem is that the shapes appearing in its statement do not have bounded treewidth. For example, the $k \times k$ grid in which the four corners are promoted to sources has treewidth k while it is a spanning subgraph of a 4-rib (by adding the diagonals of each square as well as a few edges at the sides of grid). An example is provided in Figure 7.2 when $k = 5$. In the remainder of the current section we refine Theorem 7.4.1 for graphs of treewidth at most four. First, we study the structure of hyperwebs at treewidth at most four.

7.5 Hyperwebs of treewidth at most four.

Proposition 7.5.1. *The 4-forget points of 4-hyperwebs of treewidth at most four are all anchors of the first kind.*

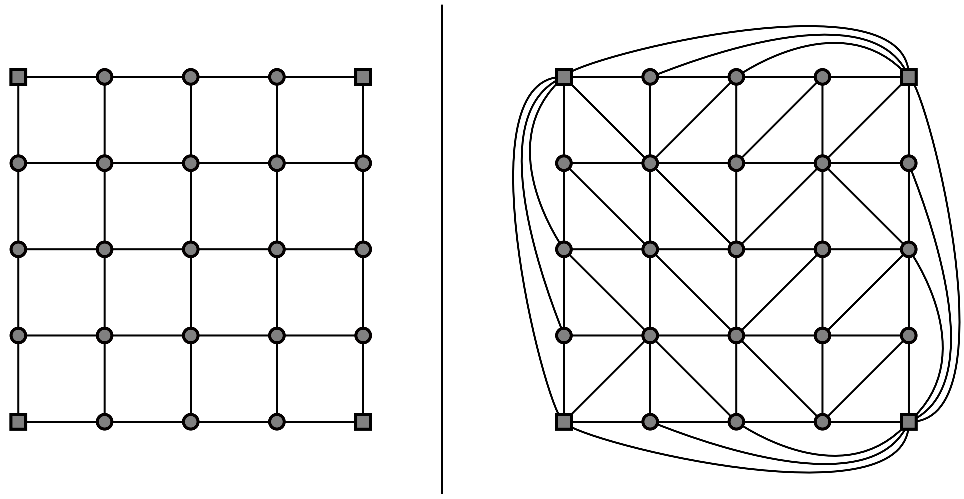


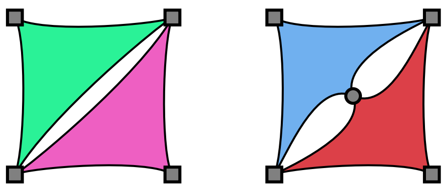
Figure 7.2: A 5×5 grid whose corners have been promoted to sources and a 4-web it is a spanning subgraph of.

Proof. Let W be a 4-hyperweb of treewidth at most four

By Proposition 4.18.2, all forget points of W are clique points or anchors.

As W is not crossed, it does not contain edges of arity five incident to all sources. Hence W does not contain clique points. By Proposition 4.2.4, the series remainder of W at any inner vertex is empty. Hence, anchors of W are of the first kind. ☐

Proposition 7.5.2. *The* 4*-hyperwebs of treewidth at most four have one of the following two shapes:*

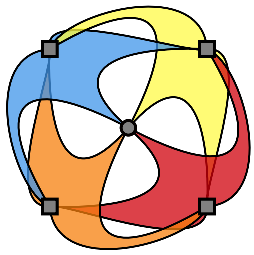


where the edges of arity four are again 4*-hyperwebs of treewidth at most four.*

Proof. The only 4-hyperweb without inner vertices is the one on the left-hand side in the statement.

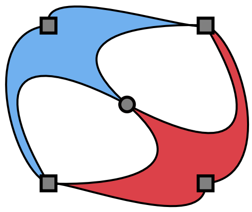
Otherwise, let W be a 4-hyperweb of treewidth at most four with an inner vertex. By Lemma 4.4.2, W has a forget point, which is an anchor by Proposition 7.5.1.

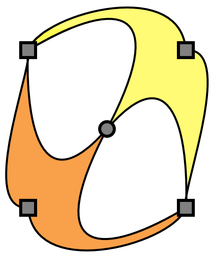
Hence, W has the following shape:



As W is a web, two consecutive edges (clock-wise) of arity four in this shape cannot both contain full prime components. Indeed, that would provide paths forming a crossing of C in W, a contradiction with Proposition 6.8.1.

Hence W has one of the following two shapes:





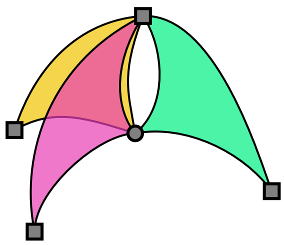
□

Proposition 7.5.3. *Every graph of treewidth at most four which has a 4-hyperweb shape has the shape of a 4-hyperweb of treewidth at most four.*

Proof. The proof works by induction using the same structure as in the proof of Proposition 7.5.2. □

7.6 K_4 sourced minors and shapes at treewidth at most four. We refine Theorem 7.4.1 for graphs of treewidth at most four. To this end, we introduce a class of hypergraphs of arity four which intuitively correspond to thickened versions of the star with three branches. We call them *star-like* graphs. Examples are provided in Figure 7.3.

Star-like graphs are built by stacking triangles, i.e. edges of arity three, on top of the following graph:



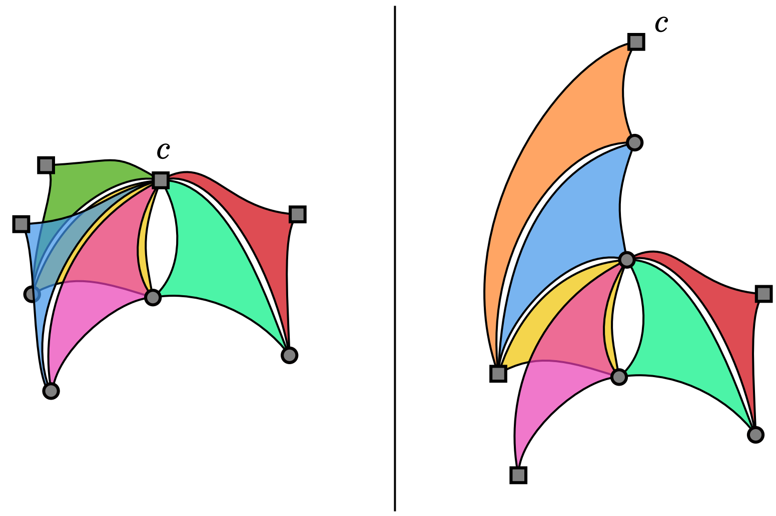
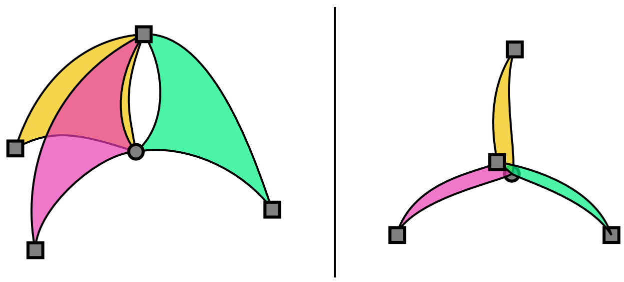


Figure 7.3: Examples of star-like graphs. The sources called c are the centres of the star-like graphs.

where, each time a triangle is added, it is incident to two sources, one of which is removed from the interface, and the third source of the triangle is added to the interface.

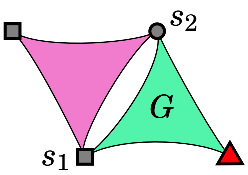
Looking at a star-like graph from above, we see the shape of a star of arity four with three branches:



The source of the star-like graph corresponding to the centre of the star with three branches in this picture is called the *centre* of the star-like graph.

We now introduce star-like graphs formally.

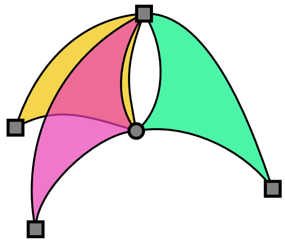
Let G be a graph and s_1 and s_2 two adjacent sources in G. We call *triangle lift* of G on s_1 and s_2 the following graph:



where the red triangular source represents sources of G distinct from s_1 and s_2.

Observe that each triangle lift replaces a source incident to the triangle by a new source.

We call a graph *star-like* if it is built by repeated triangle lifts, starting with the following graph:



In the base case of star-like graphs, one source s is adjacent to the three other sources. During the construction of a star-like graph, until a triangle lift replaces s by another source, this fact remains true.

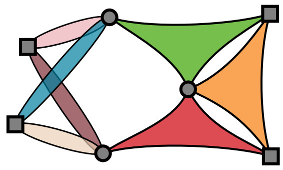
There might be several ways to construct a star-like graph. For example, the star-like graph on the left part of Figure 7.3 can be constructed by adding in any order the three triangles which are incident to two sources on top of the base case. Nonetheless, in any construction of a star-like graph, the sequence of sources replacing the source incident to all sources in the base case is identical. This fact justifies that the notion of *centre* of a star-like graph as introduced below is well-defined.

The *centre* of a star-like graph G is defined recursively as follows:

- in the base case, it is the source adjacent to all other sources, and,
- when G is obtained via a triangle lift of a star-like graph H, either the centre c_H of H is a source of G or it has been replaced in G by another source c_G. In the former case c_H is the centre of G, and in the latter case c_G is the centre of G.

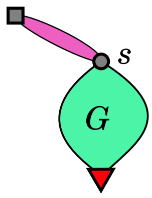
Let a be a variable of arity three. Up to adding appropriate permutations, the triangle lift of a graph G can be defined as the derived operation $\mathsf{f}(1a \parallel 1G)$. As a consequence, since the base case has treewidth four and the maximal arity of a subterm of this expression is five, star-like graphs have treewidth four.

Theorem 7.6.1. *Every graph of arity four and treewidth at most four has either K_4 as a sourced minor, or a 4-hyperweb shape, or a star-like shape, or a shape obtained via successive edge or triangle lifts of the following:*



The proof of this theorem consists in a succession of case analyses and is very long. To simplify it, we introduce a last graph operation which can be seen as a triangle lift in which the triangle is replaced by an edge of arity two.

Let G be a graph with a source s. We call *edge lift* of G on s the following graph:



where the red triangular vertex represents all sources of G distinct from s.

Lemma 7.6.2. *The edge lift of a star-like graph has a star-like shape.*

Proof. By induction on the definition of star-like graphs. □

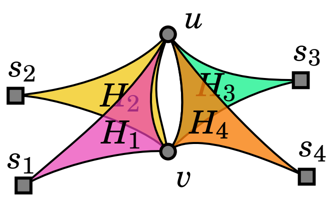
Proof of Theorem 7.6.1. Let G be a graph of arity four and treewidth at most four, which does not have K_4 as a sourced minor.

We distinguish four cases according to Theorem 7.4.1 (excluding the two leftmost shapes which are either star-like or a web).

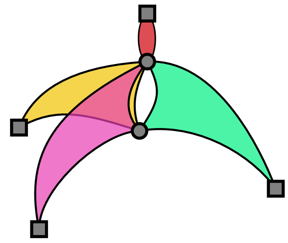
In each of these cases, G contains inner vertices. According to Proposition 4.18.2, G contains a clique point or an anchor of the first kind x. Since G does not have K_4 as a sourced minor, the series remainder at x, which can only contain atomic components of arity five by Proposition 4.2.4, is empty. Hence x is an anchor of the first kind.

Edges in shapes of G are labelled with the subgraphs substituting the edges to get G. In each case, we rely on the fact that, x being an anchor of the first kind, there is no full prime component in $G - x$.

- Assume G has the following shape:



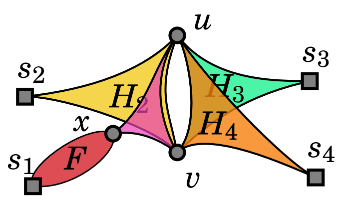
We prove by induction on $|G|$ that G has the following shape:



which has a star-like shape.

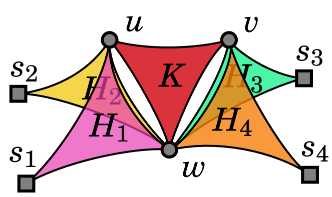
If some H_i, say H_1, does not contain paths from either u or v to s_i, then G has exactly the wanted shape.

For the sake of contradiction, assume that every H_i contains paths between the source it is incident with and both u and v, then $G - u$ and $G - v$ contain a full prime component. Hence $x \notin \{u, v\}$. If x is an inner vertex in H_i, then u and v must be disconnected from source s_i in $G - x$. Thus x is a checkpoint between s and u, and between s and v, in H_i. In other words, G has the following shape (here with $i = 1$):

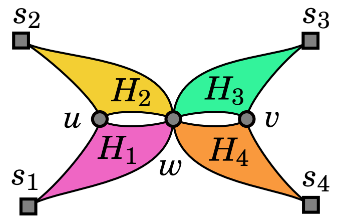


By induction hypothesis, $(G - F - s_i, x)$, and by extension G, have the wanted shape.

- Assume G has the following shape:



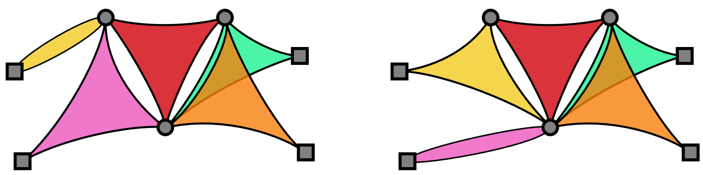
If K does not contain a path from u to v, then G has the following shape:



which has a 4-hyperweb shape.

Henceforth, assume that K contains a path from u to v.

We prove by induction on $|G|$ that G has one of the two following shapes:



which have star-like shapes.

If some H_i does not contain paths from either u, v, or w to s_i, then G has exactly one of the wanted shape.

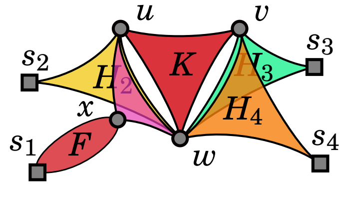
For the sake of contradiction, assume that every H_i contains paths from s_i to both u (resp. v) and w.

Thanks to K containing a path from u to v, all of $G-u$, $G-v$, and $G-w$ contain a full prime component. Hence, $x \notin \{u, v, w\}$.

Similarly, for any inner vertex y of K, $G-y$ contains a full prime component. Hence x is not an inner vertex in K.

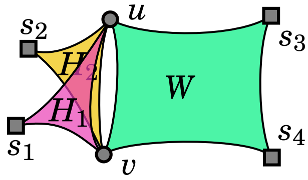
So x is inner in H_i for some $i \in \{1, 2, 3, 4\}$, say in H_1.

Since there are no full prime components in $G-x$, x must be a checkpoint in H_1 between u and s_1 and between u and s_2. In other words, G has the following shape:



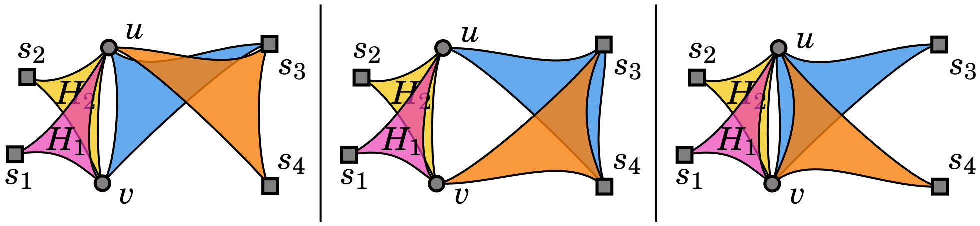
By induction hypothesis, $(G-s_1-F, x)$ has the wanted shape. This is a contradiction with the fact that all H_i contain paths from both u (resp. v) and w to s_i.

- Assume G has the following shape:



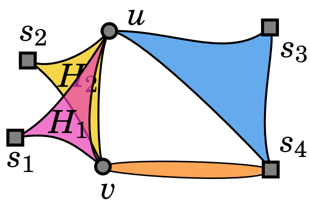
We prove, by induction on $|G|$, that G has either a 4-hyperweb shape, a star-like shape in which s_3 and s_4 are adjacent, or has a shape in which s_3 and s_4 are adjacent, obtained by successive edge and triangle lifts from the graph drawn in the statement of the theorem.

If W does not contain paths between a pair of vertices in $\{u, v, s_3, s_4\}$, then, up to swapping u and v or s_3 and s_4, G must have one of the following three shapes:



The leftmost and rightmost shapes are star-like.

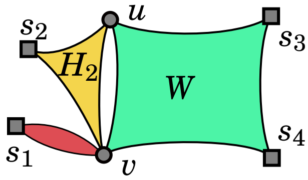
In the other case, having that W is a web ensures that us_3s_4vu is not crossed in W. Up to swapping u and v, that ensures that G has the following shape:



which has a star-like shape, and in which s_3 and s_4 are adjacent.

Henceforth we assume that W contains paths between any pair of vertices in $\{u, v, s_3, s_4\}$.

If in H_i there are no paths between s_i and either u or v (say u), then G has the following shape:



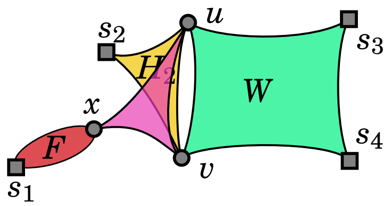
Any crossing of $s_1s_2s_3s_4s_1$ in G induces a crossing of us_3s_4vu in W. As W has a 4-hyperweb shape, such crossings do not exist, and G has a 4-hyperweb shape.

Henceforth, we assume that H_i contains paths from s_i to both u and v.

The graphs $G-u$ and $G-v$ both contain a full prime component. Hence $x \notin \{u,v\}$.

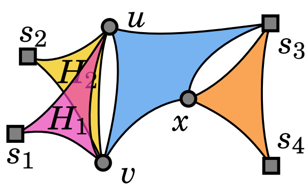
We distinguish two cases depending on whether x is in W or in H_i.

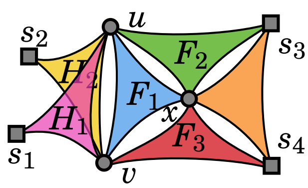
x is in H_i; Without loss of generality, assume that x is in H_1. Then x is a checkpoint in H_1 between s_1 and u and between s_1 and v. In other words, G has the following shape:



By induction hypothesis, $(G-s_1-F,x)$ has either a 4-hyperweb shape, in which case G does too, or a star-like shape in which s_3 and s_4 are adjacent, and, by Lemma 7.6.2, G has a star-like shape in which s_3 and s_4 are adjacent, or has a shape obtained by successive edge and triangle lifts from the graph drawn in the statement, in which case G does too.

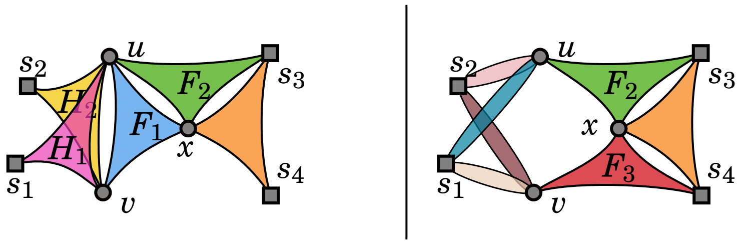
x is in W; as $G-x$ does not contain full prime components, x must be a checkpoint in W between either u or v, and either s_3 or s_4. By symmetry, we can assume that x is a checkpoint between u and s_4 and either between v and s_3 or between v and s_4. Hence G has one of the following two shapes:





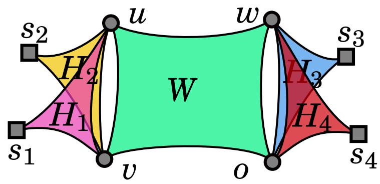
In the left shape, by induction hypothesis (this shape being a triangle lift of the case under study).

The right shape can be refined further to ensure x is an anchor of the first kind by looking at the existence of paths in F_1, F_2, and F_3, from u to v, u to s_3, and v to s_4, respectively:



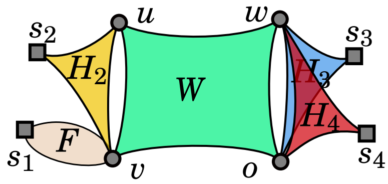
The left shape is a star-like graph and the right one is the drawn shape in the statement of the theorem, in which s_3 and s_4 are adjacent.

- Assume that G has the following shape:



We prove by induction on $|G|$ that G has either a 4-hyperweb shape, or a star-like shape, or a shape obtained by successive edge and triangle lifts from the graph drawn in the statement of the theorem.

Assume that in H_i, say H_1, there is no paths between s_1 and u or v, say v. In other words, G has shape:

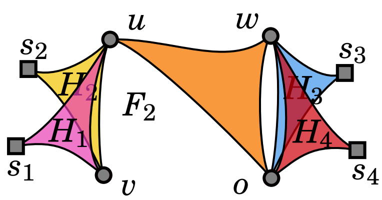


We prove that, in $(G - s_3 - s_4 - H_3 - H_4, o, w)$, $s_1 s_2 w o s_1$ is not crossed. Indeed, since v is a checkpoint between s_1 and w, a corresponding crossing would imply a crossing of $vuwov$ in W, a contradiction with W having a

4-hyperweb shape. Hence G has the shape of the previous item, and we proceed as we did there.

Assume next that all H_i contain paths between s_i and each of the two other vertices H_i is incident with.

If W does not contain a path from u or v, say v, to either w or o, then G has the following shape:



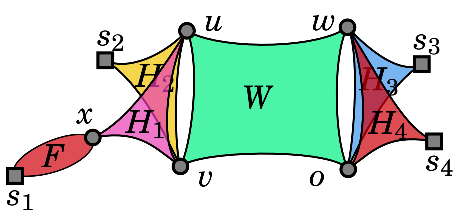
The cycle $s_1s_2s_3s_4s_1$ is crossless in G; hence G has a 4-hyperweb shape.

Henceforth, we assume that W contains a path from either u or v to either w or o.

In particular, since G contains a full prime component and H_i contains paths between s_i and each of the two vertices H_i is incident with ($i \in \{1,2,3,4\}$), all of $G-u$, $G-v$, $G-w$, and $G-o$ contain a full prime component, and $x \notin \{u,v,w,o\}$.

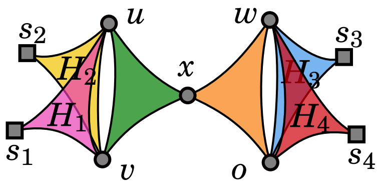
We distinguish two cases depending on whether x is in H_i or W.

x **is in** H_i**;** say $i=1$. Then x is a checkpoint in H_1 between s_1 and both u and v; G has shape



and by induction hypothesis, $(G-s_1-F, x)$ either is a spanning subgraph of a 4-hyperweb, in which case so is G, or has a star-like shape, in which case G does as well, or has a shape obtained by successive edge and triangle lifts on the graph drawn in the statement of the theorem, in which case G does as well.

x **is in** W**;** x is a checkpoint between $\{u,v\}$ and $\{w,o\}$, and G has the following shape:

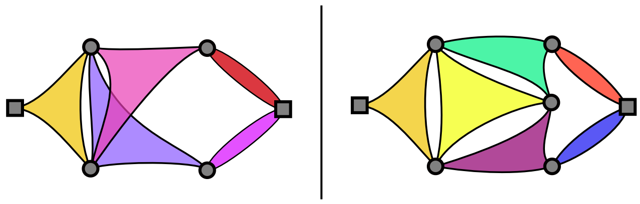


proving that $s_1s_2s_3s_4s_1$ is crossless in G; hence G has a 4-hyperweb shape. □

HARD GRAPHS OF TREEWIDTH AT MOST FOUR

7.7 Method. Our positive answer to the treewidth axiomatisation conjecture for graphs of treewidth at most three relied on a thorough analysis of hard graphs of treewidth three, which can be summarised as follows.

To obtain shapes of hard graphs to be analysed, we first proved the existence of (minimal) separation pairs (§4.26). Then, under the assumption that a hard graph of treewidth three contains two distinct separation pairs, we combined the two separation pairs, and proved that the graph has one of the following shapes (§5.4):

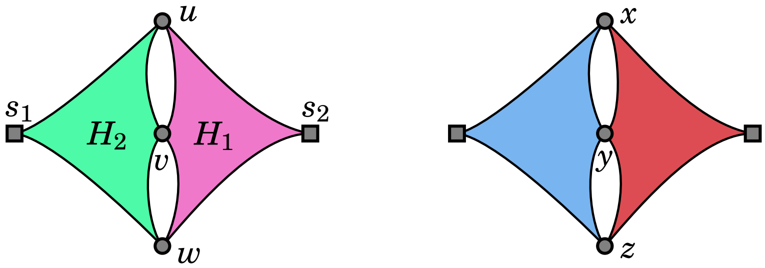


Finally, we refined these two shapes using Proposition 5.7.3 to obtain the structural property of hard graphs of treewidth at most three (Theorem 5.8.1).

Our goal in this section is to generalise this proof scheme to hard graphs of treewidth at most four. It is false that hard graphs of treewidth four always contain separation pairs or tuples. On the other hand, Proposition 5.7.3, which is used to refine the shapes of hard graphs combining separation pairs, has a generalisation in the form of Theorem 7.6.1.

We use this generalisation to explore how we can combine separation tuples in hard graphs of treewidth at most four. We chose specifically the case of separation triples in hard graphs of arity two. The reason behind our choice is that this particular case leads to a specific hard graph of treewidth at most four, that is XG_4, and the ideas behind the proof of incompleteness of C_k when $k \geq 3$ via XG_k have their origin in this incomplete exploration of hard graphs of treewidth at most four.

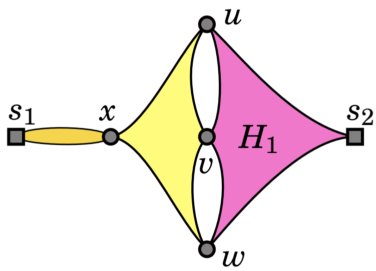
7.8 Retrieving XG_k**.** Let G be a hard graph of treewidth at most four and arity two. Suppose that G contains two disjoint minimal forget separation triples (x, y, z) and (u, v, w):



We aim to combine these two shapes. We perform only the part of the analysis that leads us to XG_4.

We distinguish several cases depending on where x, y and z are.

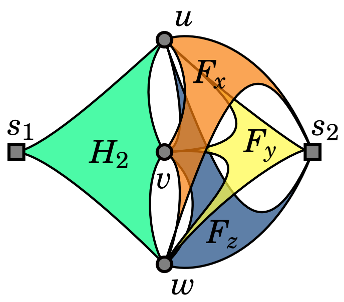
- $x \in H_2$ **and** $y, z \in H_1$**;** x must be a checkpoint in H_2 between s_1 and u, between s_1 and v, and between s_1 and w. Hence G has the following shape:



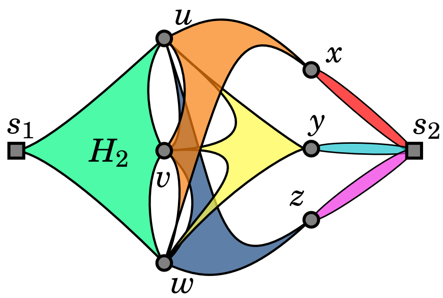
  implying that x is an anchor of G, a contradiction.

- x, y **and** z **are not all in the same graph** H_i**;** all such cases are symmetrical with the previous one.

- x, y **and** z **are all in** H_1**;** we only consider the case where x, y and z are in distinct prime components F_x, F_y, and F_z of H_1:

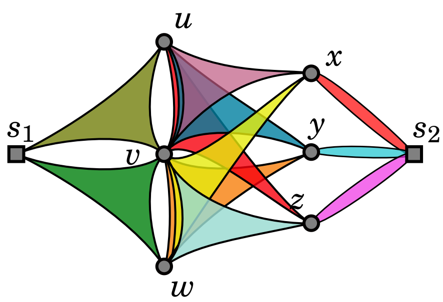


Then x, y, and z are all checkpoints between s_2 and each vertex of $\{u, v, w\}$ in their respective prime components. In other words, G has the following shape S:

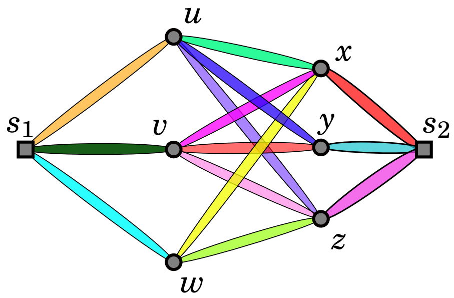


The graphs (S, x, y, z) and (S, u, v, w) have respectively treewidth four and five. According to Proposition 5.6.3, one of the k-edge in S cannot have K_k as a sourced minor.

Applying Theorem 7.6.1 and Proposition 7.5.2 repeatedly, each time replacing 4-edges by simple webs, and choosing the unique web with only four vertices, yields the following shape S':



The graph (S', u, v, w) has treewidth five. Keep applying Proposition 5.6.3, but with Proposition 5.7.3 and we get XG_4:



These ideas led to the general definition of XG_k and the proof of Proposition 4.21.2.

7.9 Conclusion. The starting point of this thesis is the treewidth axiomatisation conjecture as expressed by Courcelle and Engelfriet [12, p. 118].

Several problems occur: finding an appropriate syntax of terms for graphs of bounded treewidth, finding interesting axioms sets, and then decomposing graphs of bounded treewidth to prove the completeness of the axiom sets.

Our syntax, described in §3.9, has specificities that simplify basic graph decompositions, such as in full prime components (§4.2). In contrast with the syntax of Courcelle and Engelfriet, we only allow parallel compositions of graphs of the same arities.

We are able to decompose graphs of bounded treewidth in full prime components and along their anchors and to follow these decompositions on terms, using axioms (see Chapter 4). This allows us to provide a positive answer to the treewidth axiomatisation conjecture for graphs of treewidth at most k, when $k \leq 2$. However, this is not sufficient in general, and our answer when $k = 3$ uses a detailed study of hard graphs of treewidth at most three (see Chapter 5).

Attempts to generalise from $k = 3$ to $k = 4$ led us to the study of the two paths theorem [43, 38, 34, 24]. We have provided a new proof of this theorem from which we extracted an algorithm for the two disjoint paths problem (see Chapter 6). Our algorithm contrasts by its simplicity with the more efficient but involved algorithms existing in the literature.

The two paths theorem is used in [29] to characterise graphs of arity four without K_4 as a sourced minor. In the current chapter, we generalised this result to general hypergraphs, and to hypergraphs of treewidth at most four. We regard this result as a generalisation of Proposition 5.7.3 which is a cornerstone in our analysis of hard graphs of treewidth at most three, and we regard it as a promising starting point towards a positive answer to the treewidth axiomatisation conjecture for graphs of treewidth at most four.

However, this is not sufficient since we know that hard graphs of treewidth at most four do not necessarily contain separation tuples (see Figure 4.10), and even if they did, combining distinct separation pairs and/or triples in hard graphs of treewidth four seems to require more case analysis than in the already long proof combining separation pairs in hard graphs of treewidth at most three (Proposition 5.4.1).

An interesting consequence of our structural property of hard graphs of treewidth at most three (Theorem 5.8.1) is the well-known structural characterisation of graphs of treewidth at most two as graphs without K_4 as a minor (§5.10). For any non-negative integer k, we believe that the study of hard graphs of treewidth at most $k+1$ in general could lead to a better understanding of the precise minimal excluded minors characterising graphs of treewidth at most k.

Contents (detailed)

Bibliography

[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] Stefan Arnborg and Andrzej Proskurowski. Canonical representations of partial 2-and 3-trees. *BIT Numerical Mathematics*, 32(2):197–214, 1992.

[4] Stefan Arnborg, Andrzej Proskurowski, and Derek G. Corneil. Forbidden minors characterization of partial 3-trees. *Discret. Math.*, 80(1):1–19, 1990. doi:10.1016/0012-365X(90)90292-P.

[5] Julien Bensmail, Romain Bourneuf, Paul Colinot, Samuel Humeau, and Timothée Martinod. Making graphs irregular through irregularising walks. *CoRR*, abs/2506.21254, 2025. arXiv:2506.21254, doi:10.48550/ARXIV.2506.21254.

[6] 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.

[7] 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.

[8] Johannes Carmesin and Jan Kurkofka. Canonical decompositions of 3-connected graphs. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1887–1920. IEEE, 2023. doi:10.1109/FOCS57990.2023.00115.

[9] Gary Chartrand and Ping Zhang. *A first course in graph theory*. Dover Publications, 2012.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

[17] Jens Gustedt. The general two-path problem in time O(m log n). Technical report, INRIA, 1994.

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

[19] John E. Hopcroft and Robert E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.

[20] Samuel Humeau, Daniela Petrisan, and Jurriaan Rot. Correspondences between codensity and coupling-based liftings, a practical approach. In Jörg Endrullis and Sylvain Schmitz, editors, *33rd EACSL Annual Conference on Computer Science Logic, CSL 2025, February 10-14, 2025, Amsterdam, Netherlands*, volume 326 of *LIPIcs*, pages 29:1–29:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.CSL.2025.29.

[21] Samuel Humeau and Damien Pous. On the two paths theorem and the two disjoint paths problem. *CoRR*, abs/2505.16431, 2025. arXiv:2505.16431, doi:10.48550/ARXIV.2505.16431.

[22] Heinz A Jung. Eine verallgemeinerung des n-fachen zusammenhangs für graphen. *Mathematische Annalen*, 187(2):95–103, 1970.

[23] Ken-ichi Kawarabayashi, Zhentao Li, and Bruce Reed. Connectivity preserving iterative compaction and finding 2 disjoint rooted paths in linear time. *arXiv preprint arXiv:1509.07680*, 2015.

[24] Ken-ichi Kawarabayashi, Robin Thomas, and Paul Wollan. A new proof of the flat wall theorem. *Journal of Combinatorial Theory, Series B*, 129:204–238, 2018.

[25] Richard Kaye. *Models of Peano Arithmetic*. Oxford Logic Guides 15. Oxford University Press, USA, 1991.

[26] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. Processor efficient parallel algorithms for the two disjoint paths problem and for finding a kuratowski homeomorph. *SIAM J. Comput.*, 21(3):486–506, 1992. doi:10.1137/0221032.

[27] Jan Kurkofka and Tim Planken. A tutte-type canonical decomposition of 4-connected graphs. *CoRR*, abs/2504.00760, 2025. arXiv:2504.00760, doi:10.48550/ARXIV.2504.00760.

[28] Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM Journal on Computing*, 46(1):161–189, 2017.

[29] Ruy Fabila Monroy and David R. Wood. Rooted k_4-minors. *Electron. J. Comb.*, 20(2):64, 2013. doi:10.37236/3476.

[30] Yehoshua Perl and Yossi Shiloach. Finding two disjoint paths between two pairs of vertices in a graph. *J. ACM*, 25(1):1–9, 1978. doi:10.1145/322047.322048.

[31] Damien Pous. Hypergraphs, November 2024. Accessed on 2025-08-19. doi:10.4230/ARTIFACTS.22445.

[32] Neil Robertson and Paul D. Seymour. Graph minors i–xxiii. *J. Combin. Theory Ser. B*, 1983–2010.

[33] 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.

[34] Neil Robertson and Paul D. Seymour. Graph minors. ix. disjoint crossed paths. *Journal of Combinatorial Theory Series B*, 49(1):40–77, 1990.

[35] 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.

[36] Appajosyula Satyanarayana and L. Tung. A characterization of partial 3-trees. *Networks*, 20(3):299–322, 1990. doi:10.1002/NET.3230200304.

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

[38] Paul D. Seymour. Disjoint paths in graphs. *Discret. Math.*, 29(3):293–309, 1980. doi:10.1016/0012-365X(80)90158-2.

[39] Yossi Shiloach. A polynomial solution to the undirected two paths problem. *J. ACM*, 27(3):445–456, 1980. doi:10.1145/322203.322207.

[40] John W. Suurballe and Robert E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984. doi:10.1002/net.3230140209.

[41] Torsten Tholey. Solving the 2-disjoint paths problem in nearly linear time. *Theory Comput. Syst.*, 39(1):51–78, 2006. doi:10.1007/S00224-005-1256-9.

[42] Torsten Tholey. Improved algorithms for the 2-vertex disjoint paths problem. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 546–557. Springer, 2009.

[43] Carsten Thomassen. 2-linked graphs. *Eur. J. Comb.*, 1(4):371–378, 1980. doi:10.1016/S0195-6698(80)80039-4.

[44] William Thomas Tutte. Convex representations of graphs. In *Proc. London Math. Soc.*, volume 10, pages 304–320, 1960. doi:10.1112/plms/s3-10.1.304.

[45] William Thomas Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966. URL: http://www.jstor.org/stable/10.3138/j.ctvfrxhc8.

[46] William Thomas Tutte. On the enumeration of convex polyhedra. *Journal of Combinatorial Theory, Series B*, 28(2):105–126, 1980. doi:10.1016/0095-8956(80)90059-3.

[47] Mark E. Watkins. On the existence of certain disjoint arcs in graphs. *Duke Mathematical Journal*, 35(2):231 – 246, 1968. doi:10.1215/S0012-7094-68-03523-0.

[48] Wolfgang Wechler. *Universal algebra for computer scientists*, volume 25. Springer Science & Business Media, 2012.

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

[50] David R. Wood and Svante Linusson. Thomassen's choosability argument revisited. *SIAM J. Discret. Math.*, 24(4):1632–1637, 2010. doi:10.1137/100796649.