Logic, Graphs, and Algorithms

Martin Grohe

Humboldt-Universität zu Berlin

September 10, 2007

Abstract

Algorithmic meta theorems are algorithmic results that apply to whole families of combinatorial problems, instead of just specific problems. These families are usually defined in terms of logic and graph theory. An archetypal algorithmic meta theorem is Courcelle's Theorem [9], which states that all graph properties definable in monadic second-order logic can be decided in linear time on graphs of bounded tree width.

This article is an introduction into the theory underlying such meta theorems and a survey of the most important results in this area.

1 Introduction

In 1990, Courcelle [9] proved a fundamental theorem stating that graph properties definable in monadic second-order logic can be decided in linear time on graphs of bounded tree width. This is the first in a series of *algorithmic meta theorems*. More recent examples of such meta theorems state that all first-order definable properties of planar graphs can be decided in linear time [43] and that all first-order definable optimisation problems on classes of graphs with excluded minors can be approximated in polynomial time to any given approximation ratio [19]. The term "meta theorem" refers to the fact that these results do not describe algorithms for specific problems, but for whole families of problems, whose definition typically has a logical and a structural (usually graph theoretical) component. For example, Courcelle's Theorem is about *monadic second-order logic* on *graphs of bounded tree width*.

This article is an introductory survey on algorithmic meta theorems. Why should we care about such theorems? First of all, they often provide a quick way to prove that a problem is solvable efficiently. For example, to show that the 3-colourability problem can be solved in linear time on graphs of bounded tree width, we observe that 3-colourability is a property of graphs definable in monadic second-order logic and apply Courcelle's theorem. Secondly, and more substantially, algorithmic meta theorems yield a better understanding of the scope of general algorithmic techniques and, in some sense, the limits of tractability. In particular, they clarify the interactions between logic and combinatorial structure, which is fundamental for computational complexity.

The general form of algorithmic meta theorems is:

> All *problems* definable in a certain *logic* on a certain class of *structures* can be solved *efficiently*.

Problems may be of different types, for example, they may be optimisation or counting problems, but in this article we mainly consider decision problems. We briefly discuss other types of problems in Section 7.2. *Efficient solvability* may mean, for example, polynomial time solvability, linear or quadratic time solvability, or fixed-parameter tractability. We will discuss this in detail in Section 2.3. Let us now focus on the two main ingredients of the meta theorems, logic and structure.

The two *logics* that, so far, have been considered almost exclusively for meta theorems are first-order logic and monadic second-order logic. Techniques from logic underlying the theorems are Feferman-Vaught style composition lemmas, automata theoretic techniques, and locality results such as Hanf's Theorem and Gaifman's Theorem.

The *structures* in algorithmic meta theorems are usually defined by graph theoretic properties. Actually, to ease the presentation, the only structures we will consider in this survey are graphs. Many of the meta theorems are tightly linked with *graph minor theory*. This deep theory, mainly developed by Robertson and Seymour in a long series of papers, describes the structure of graphs with excluded minors. It culminates in the graph minor theorem [73], which states that every class of graphs closed under taking minors can be characterised by a finite set of excluded minors. The theory also has significant algorithmic consequences. Robertson and Seymour [71] proved that every class of graphs that is closed under taking minors can be recognised in cubic time. More recently, results from graph minor theory have been combined with algorithmic techniques that had originally been developed for planar graphs to obtain polynomial time approximation schemes and fixed parameter tractable algorithms for many standard optimisation problems on families of graphs with excluded minors. The methods developed in this context are also underlying the more advanced algorithmic meta theorems.

There are some obvious similarities between algorithmic meta theorems and results from *descriptive complexity theory*, in particular such results from descriptive complexity theory that also involve restricted classes of graphs. As an example, consider the theorem stating that fixed-point logic with counting captures polynomial time on graphs of bounded tree width [49], that is, a property of graphs of bounded tree width is definable in fixed-point logic with counting if and only if it is decidable in polynomial time. Compare this to Courcelle's Theorem. Despite the similarity, there are two crucial differences: On the one hand, Courcelle's Theorem is weaker as it makes no completeness claim, that is, it does not state that *all* properties of graphs of bounded tree width that are decidable in linear time are definable in monadic second-order logic. On the other hand, Courcelle's Theorem is stronger in its algorithmic content. Whereas it is very easy to show that all properties of graphs (not only graphs of bounded tree width) definable in fixed-point logic with counting are decidable in polynomial time, the proof of Courcelle's theorem relies on substantial algorithmic ideas like the translation of monadic second-order logic over trees into tree automata [78] and a linear time algorithm for computing tree decompositions [5]. In general, algorithmic meta theorems involve nontrivial algorithms, but do not state completeness, whereas in typical results from descriptive complexity, the algorithmic content is limited, and the nontrivial part is completeness. But there is no clear dividing line. Consider, for example, Papadimitriou and Yannakakis's [64] well known result that all optimisation problems in the logically defined class MAXSNP have a constant factor approximation algorithm. This theorem does not state completeness, but technically it is much closer to Fagin's Theorem [37], a central result of descriptive complexity theory, than to the algorithmic meta theorems considered here. In any case, both algorithmic meta theorems and descriptive complexity theory are branches of finite model theory, and there is no need to draw a line between them.

When I wrote this survey, it was my goal to cover the developments up to the most recent and strongest results, which are concerned with monadic second-order logic on graphs of bounded rank width and with first-order logic on graphs with excluded minors. The proofs of most theorems are at least sketched, so that hopefully the reader will not only get an impression of the results, but also of the techniques involved in their proofs.

2 The basics

$\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$, and $\mathbb{N}$ denote the sets of real numbers, rational numbers, integers, and natural numbers (that is, positive integers), respectively. For a set $S \subseteq \mathbb{R}$, by $S_{\geq 0}$ we denote the set of nonnegative numbers in S. For integers m, n, by $[m, n]$ we denote the interval $\{m, m+1, \ldots, n\}$, which is empty if $n < m$. Furthermore, we let $[n] = [1, n]$. The power set of a set S is denoted by 2^S, and the set of all k-element subsets of S by $\binom{S}{k}$.

2.1 Graphs

A *graph* G is a pair $(V(G), E(G))$, where $V(G)$ is a finite set whose elements are called *vertices* and $E(G) \subseteq \binom{V(G)}{2}$ is a set of unordered pairs of vertices, which are called *edges*. Hence graphs in this paper

are always *finite*, *undirected*, and *simple*, where simple means that there are no loops or parallel edges. If $e = \{u,v\}$ is an edge, we say that the vertices u and v are *adjacent*, and that both u and v are *incident* with e. A graph H is a *subgraph* of a graph G (we write $H \subseteq G$) if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If $E(H) = E(G) \cap \binom{V(H)}{2}$, then H is an *induced* subgraph of G. For a set $W \subseteq V(G)$, we write $G[W]$ to denote the induced subgraph $\left(W, E(G) \cap \binom{W}{2}\right)$ and $G \setminus W$ to denote $G[V(G) \setminus W]$. For a set $F \subseteq E$, we let $G[\![F]\!]$ be the subgraph $\left(\bigcup F, F\right)$. Here $\bigcup F$ denote the union of all edges in F, that is, the set of all vertices incident with at least one edge in F. We call $G[\![F]\!]$ the subgraph of G *generated* by F; note that it is not necessarily an induced subgraph of G. The *union* of two graphs G and H is the graph $G \cup H = (V(G) \cup V(H), E(G) \cup E(H))$, and the *intersection* $G \cap H$ is defined similarly. The *complement* of a graph $G = (V,E)$ is the graph $\overline{G} = \left(V, \binom{V}{2} \setminus E\right)$. There is a unique *empty graph* $(\emptyset, \emptyset)$. For $n \geq 1$, we let K_n be the complete graph with n vertices. To be precise, let us say $K_n = \left([n], \binom{[n]}{2}\right)$. We let $K_{n,m}$ be the complete bipartite graph with parts of size m, n, respectively.

Occasionally, we consider (vertex) labelled graphs. A *labelled graph* is a tuple

$$G = \big(V(G), E(G), P_1(G), \ldots, P_\ell(G)\big),$$

where $P_i(G) \subseteq V(G)$ for all $i \in [\ell]$. The symbols P_i are called *labels*, and if $v \in P_i(G)$ we say that v is *labelled* by P_i. Subgraphs, union, and intersection extend to labelled graphs in a straightforward manner. The *underlying graph* of a labelled graph G is $(V(G), E(G))$. Whenever we apply graph theoretic notions such as connectivity to labelled graphs, we refer to the underlying graph.

The *order* $|G|$ of a graph G is the number of vertices of G. We usually use the letter n to denote the order of a graph. The *size* of G is the number $||G|| = |G| + |E(G)|$. Up to a constant factor, this is the size of the adjacency list representation of G under a uniform cost model.

$\mathscr{G}$ denotes the class of all graphs. For every class $\mathscr{C}$ of graphs, we let $\mathscr{C}_{lb}$ be the class of all labelled graphs whose underlying graph is in $\mathscr{C}$. A *graph invariant* is a mapping defined on the class $\mathscr{G}$ of all graphs that is invariant under isomorphisms. All graph invariants considered in this paper are integer valued. For a graph invariant $f : \mathscr{G} \to \mathbb{Z}$ and a class $\mathscr{C}$ of graphs, we say that $\mathscr{C}$ *has bounded* f if there is a $k \in \mathbb{Z}$ such that $f(G) \leq k$ for all $G \in \mathscr{C}$.

Let $G = (V,E)$ be a graph. The *degree* $\deg^G(v)$ of a vertex $v \in V$ is the number of edges incident with v. We omit the superscript ^{G} if G is clear from the context. The *(maximum) degree* of G is the number

$$\Delta(G) = \max\{\deg(v) \mid v \in V\}.$$

The *minimum degree* $\delta(G)$ is defined analogously, and the *average degree* $d(G)$ is $2|E(G)|/|V(G)|$. Observe that $||G|| = O(d(G) \cdot |G|)$. Hence if a class $\mathscr{C}$ of graphs has bounded average degree, then the size of the graphs in $\mathscr{C}$ is linearly bounded in the order. In the following, "degree" of a graph, without qualifications, always means "maximum degree".

A *path* in $G = (V,E)$ of *length* $n \geq 0$ *from* a vertex v_0 *to* a vertex v_n is a sequence $v_0, \ldots, v_n$ of distinct vertices such that $\{v_{i-1}, v_i\} \in E$ for all $i \in [n]$. Note that the length of a path is the number of edges on the path. Two paths are *disjoint* if they have no vertex in common. G is *connected* if it is nonempty and for all $v, w \in V$ there is a path from v to w. A *connected component* of G is a maximal (with respect to $\subseteq$) connected subgraph. G is *k-connected*, for some $k \geq 1$, if $|V| > k$ and for every $W \subseteq V$ with $|W| < k$ the graph $G \setminus W$ is connected.

A *cycle* in a graph $G = (V,E)$ of *length* $n \geq 3$ is a sequence $v_1 \ldots v_n$ of distinct vertices such that $\{v_n, v_1\} \in E$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in [2,n]$. A graph G is *acyclic*, or a *forest*, if it has no cycle. G is a *tree* if it is acyclic and connected. It will be a useful convention to call the vertices of trees *nodes*. A node of degree at most 1 is called a *leaf*. The set of all leaves of a tree T is denoted by $L(T)$. Nodes that are not leaves are called *inner nodes*. A *rooted tree* is a triple $T = (V(T), E(T), r(T))$, where $(V(T), E(T))$ is a tree and $r(T) \in V(T)$ is a distinguished node called the *root*. A node t of a rooted tree T is the *parent* of a node u, and u is a *child* of t, if t is the predecessor of u on the unique path from the root $r(T)$ to u. Two nodes that are children of the same parent are called *siblings*. A *binary tree* is a rooted tree T in which every node has either no children at all or exactly two children.

3

2.2 Logic

I assume that the reader has some background in logic and, in particular, is familiar with first-order predicate logic. To simplify matters, we only consider logics over (labelled) graphs, even though most results mentioned in this survey extend to more general structures. Let us briefly review the syntax and semantics of *first-order logic* FO and *monadic second-order logic* MSO. We assume that we have an infinite supply of *individual variables*, usually denoted by the lowercase letters x, y, z, and an infinite supply of *set variables*, usually denoted by uppercase letters X, Y, Z. *First-order formulas* in the language of graphs are built up from atomic formulas $E(x, y)$ and $x = y$ by using the usual Boolean connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (bi-implication) and existential quantification $\exists x$ and universal quantification $\forall x$ over individual variables. Individual variables range over vertices of a graph. The atomic formula $E(x, y)$ expresses adjacency, and the formula $x = y$ expresses equality. From this, the semantics of first-order logic is defined in the obvious way. First-order formulas over labelled graphs may contain additional atomic formulas $P_i(x)$, meaning that x is labelled by P_i. If a label P_i does not appear in a labelled graph G, then we always interpret $P_i(G)$ as the empty set. In *monadic second-order formulas*, we have additional atomic formulas $X(x)$ for set variables X and individual variables x, and we admit existential and universal quantification over set variables. Set variables are interpreted by sets of vertices, and the atomic formula $X(x)$ means that the vertex x is contained in the set X.

The *free* individual and set variables of a formula are defined in the usual way. A *sentence* is a formula without free variables. We write $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ to indicate that φ is a formula with free variables among $x_1, \ldots, x_k, X_1, \ldots, X_\ell$. We use this notation to conveniently denote substitutions and assignments to the variables. If $G = (V, E)$ is a graph, $v_1, \ldots, v_k \in V$, and $W_1, \ldots, W_\ell \subseteq V$, then we write $G \models \varphi(v_1, \ldots, v_k, W_1, \ldots, W_\ell)$ to denote that $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ holds in G if the variables x_i are interpreted by the vertices v_i and the variables X_i are interpreted by the vertex sets W_i.

Occasionally, we consider monadic second-order formulas that contain no second-order quantifiers, but have free set variables. We view such formulas as first-order formulas, because free set variables are essentially the same as labels (unary relation symbols). An example of such a formula is the formula $dom(X)$ in Example 2.1 below. We say that a formula $\varphi(X)$ is *positive in X* if X only occurs in the scope of an even number of negation symbols. It is *negative in X* if X only occurs in the scope of an odd number of relation symbols.

We freely use abbreviations such as $\bigwedge_{i=1}^{k} \varphi_i$ instead of $(\varphi_1 \wedge \ldots \wedge \varphi_k)$ and $x \neq y$ instead of $\neg x = y$.

Example 2.1. A *dominating set* in a graph $G = (V, E)$ is a set $S \subseteq V$ such that for every $v \in V$, either v is in S or v is adjacent to a vertex in S.

The following first-order sentence dom_k says that a graph has a dominating set of size k:

$$dom_k = \exists x_1 \ldots \exists x_k \Big(\bigwedge_{1 \le i < j \le k} x_i \neq x_j \wedge \forall y \bigvee_{i=1}^{k} \big(y = x_i \vee E(y, x_i) \big) \Big).$$

The following formula $dom(X)$ says that X is a dominating set:

$$dom(X) = \forall y \Big(X(y) \vee \exists z \big(X(z) \wedge E(z, y) \big) \Big).$$

More precisely, for every graph G and every subset $S \subseteq V(G)$ it holds that $G \models dom(S)$ if and only if S is a dominating set of G. ⌟

Example 2.2. The following monadic second-order sentences *conn* and *acyc* say that a graph is connected and acyclic, respectively:

$$conn = \exists x\, x = x \wedge \forall X \Big(\big(\exists x X(x) \wedge \forall x \forall y \big((X(x) \wedge E(x, y)) \rightarrow X(y) \big) \big) \rightarrow \forall x X(x) \Big),$$

$$acyc = \neg \exists X \Big(\exists x X(x) \wedge \forall x \big(X(x) \rightarrow \exists y_1 \exists y_2 \big(y_1 \neq y_2 \wedge E(x, y_1) \wedge E(x, y_2) \wedge X(y_1) \wedge X(y_2) \big) \big) \Big).$$

The sentence *acyc* is based on the simple fact that a graph has a cycle if and only if it has a nonempty induced subgraph in which every vertex has degree at least 2. Then the sentence *tree = conn ∧ acyc* says that a graph is a tree. ⌟
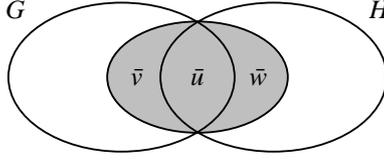
4

Figure 2.1. An illustration of Lemma 2.3

The *quantifier rank* of a first-order or monadic second-order formula φ is the nesting depth of quantifiers in φ. For example, the quantifier rank of the formula *acyc* in Example 2.2 is 4. Let G be a graph and $\bar{v} = (v_1, \ldots, v_k) \in V(G)^k$, for some nonnegative integer k. For every $q \geq 0$, the *first-order q-type of $\bar{v}$ in G* is the set $\mathrm{tp}_q^{\mathrm{FO}}(G, \bar{v})$ of all first-order formulas $\varphi(x_1, \ldots, x_k)$ of quantifier rank at most q such that $G \models \varphi(v_1, \ldots, v_k)$. The *monadic second-order q-type of $\bar{v}$ in G*, $\mathrm{tp}_q^{\mathrm{MSO}}(G, \bar{v})$ is defined analogously. As such, types are infinite sets, but we can syntactically *normalise* formulas in such a way that there are only finitely many normalised formulas of fixed quantifier rank and with a fixed set of free variables, and that every formula can effectively be transformed into an equivalent normalised formula of the same quantifier rank. We represent a type by the set of normalised formulas it contains. There is a fine line separating decidable and undecidable properties of types and formulas. For example, it is decidable whether a formula is contained in a type: We just normalise the formula and test if it is equal to one of the normalised formulas in the type. It is undecidable whether a set of normalised formulas actually is (more precisely: represents) a type. To see this, remember that types are satisfiable by definition and that the satisfiability of first-order formulas is undecidable.

For a tuple $\bar{v} = (v_1, \ldots, v_k)$, we sloppily write $\{\bar{v}\}$ to denote the set $\{v_1, \ldots, v_k\}$. It will always be clear from the context whether $\{\bar{v}\}$ refers to the set $\{v_1, \ldots, v_k\}$ or the 1-element set $\{(v_1, \ldots, v_k)\}$. For tuples $\bar{v} = (v_1, \ldots, v_k)$ and $\bar{w} = (w_1, \ldots, w_\ell)$, we write $\bar{v}\bar{w}$ to denote their concatenation $(v_1, \ldots, v_k, w_1, \ldots, w_\ell)$. We shall heavily use the following "Feferman-Vaught style" composition lemma.

Lemma 2.3. *Let* tp *be one of* $\mathrm{tp}^{\mathrm{FO}}, \mathrm{tp}^{\mathrm{MSO}}$. *Let G, H be labelled graphs and $\bar{u} \in V(G)^k$, $\bar{v} \in V(G)^\ell$, $\bar{w} \in V(H)^m$ such that $V(G) \cap V(H) = \{\bar{u}\}$ (cf. Figure 2.1). Then for all $q \geq 0$, $\mathrm{tp}_q(G \cup H, \bar{u}\bar{v}\bar{w})$ is determined by $\mathrm{tp}_q(G, \bar{u}\bar{v})$ and $\mathrm{tp}_q(H, \bar{u}\bar{w})$. Furthermore, there is an algorithm that computes $\mathrm{tp}_q(G \cup H, \bar{u}\bar{v}\bar{w})$ from $\mathrm{tp}_q(G, \bar{u}\bar{v})$ and $\mathrm{tp}_q(H, \bar{u}\bar{w})$.*

Let me sketch a proof of this lemma for first-order types. The version for monadic second-order types can be proved similarly, but is more complicated (see, for example, [57]).

Proof sketch. Let G, H be labelled graphs and $\bar{u} \in V(G)^k$ such that $V(G) \cap V(H) = \{\bar{u}\}$. By induction on φ, we prove the following claim:

Claim: Let $\varphi(\bar{x}, \bar{y}, \bar{z})$ be a first-order formula of quantifier rank q, where $\bar{x}$ is a k-tuple and $\bar{y}$, $\bar{z}$ are tuples of arbitrary length. Then there is a Boolean combination $\Phi(\bar{x}, \bar{y}, \bar{z})$ of expressions $G \models \psi(\bar{x}, \bar{y})$ and $H \models \chi(\bar{x}, \bar{z})$ for formulas ψ, χ of quantifier rank at most q, such that for all tuples $\bar{v}$ of vertices of G and $\bar{w}$ of vertices of H of the appropriate lengths it holds that

$$G \cup H \models \varphi(\bar{u}, \bar{v}, \bar{w}) \iff \Phi(\bar{u}, \bar{v}, \bar{w}).$$

Here $\Phi(\bar{u}, \bar{v}, \bar{w})$ denotes the statement obtained from $\Phi(\bar{x}, \bar{y}, \bar{z})$ by substituting $\bar{u}$ for $\bar{x}$, $\bar{v}$ for $\bar{y}$, and $\bar{w}$ for $\bar{z}$.

Furthermore, the construction of Φ from φ is effective.

The claim holds for atomic formulas, because there are no edges from $V(G) \setminus V(H)$ to $V(H) \setminus V(G)$ in $G \cup H$. It obviously extends to Boolean combinations of formulas. So suppose that $\varphi(\bar{x}, \bar{y}, \bar{z}) = \exists x_0 \psi(\bar{x}, x_0, \bar{y}, \bar{z})$. Let $\bar{v}, \bar{w}$ be tuples in G, H of the appropriate lengths. By the induction hypothesis, there are $\Psi_1(\bar{x}, \bar{y}y_0, \bar{z})$ and $\Psi_2(\bar{x}, \bar{y}, \bar{z}z_0)$ such that

$$\begin{aligned} &G \cup H \models \varphi(\bar{u}, \bar{v}, \bar{w}) \\ \iff &\exists v_0 \in V(G)\ \Psi_1(\bar{u}, \bar{v}v_0, \bar{w}) \text{ or } \exists w_0 \in V(H)\ \Psi_2(\bar{u}, \bar{v}, \bar{w}w_0). \end{aligned}$$

We may assume that Ψ_1 is of the form

$$\bigvee_{i=1}^{m} \big(G \models \chi_i(\bar{x}, \bar{y}y_0) \wedge H \models \xi_i(\bar{x}, \bar{z})\big).$$

Hence $\exists v_0 \in V(G)\,\Psi_1(\bar{u}, \bar{v}v_0, \bar{w})$ is equivalent to

$$\bigvee_{i=1}^{m} \big(\exists v_0 \in V(G)\ G \models \chi_i(\bar{u}, \bar{v}v_0) \wedge H \models \xi_i(\bar{u}, \bar{w})\big).$$

We let $\Phi_1 = \bigvee_{i=1}^{m} \big(G \models \exists y_0 \chi_i(\bar{x}, \bar{y}y_0) \wedge H \models \xi_i(\bar{x}, \bar{z})\big)$. Similarly, we define a Φ_2 from Ψ_2, and then we let $\Phi = \Phi_1 \vee \Phi_2$.

Clearly, the claim implies the statements of the lemma. $\qquad\square$

2.3 Algorithms and complexity

I assume that the reader is familiar with the basics of the design and analysis of algorithms. We will make extensive use of the Oh-notation. For example, we will denote the class of all polynomially bounded functions of one variable n by $n^{O(1)}$. I also assume that the reader is familiar with standard complexity classes such as PTIME, NP, and PSPACE and with concepts such as reducibility between problems and hardness and completeness for complexity classes. If not specified otherwise, reductions are always polynomial time many-one reductions. The following example illustrates our notation for introducing algorithmic problems.

Example 2.4. The *dominating set problem* is defined as follows:

DOMINATING-SET
Instance: A graph G and a natural number k.
Problem: Decide if G has a dominating set of size k.

It is well-known that DOMINATING-SET is NP-complete. ⌐

We are mainly interested in algorithms for and the complexity of *model checking problems*. For every logic L and every class $\mathscr{C}$ of graphs, we let:

MC(L, $\mathscr{C}$)
Instance: A sentence φ of L and a graph $G \in \mathscr{C}$.
Problem: Decide if $G \models \varphi$.

If $\mathscr{C}$ is the class of all graphs, we just write MC(L).

Example 2.5. Example 2.1 shows that DOMINATING-SET is reducible to MC(FO). Hence MC(FO) is NP-hard. As MC(FO) is trivially reducible to MC(MSO), the latter is also NP-hard. ⌐

Fact 2.6 (Vardi [79]). MC(FO) *and* MC(MSO) *are* PSPACE-*complete.*

This fact is often phrased as: "The *combined complexity* of FO resp. MSO is PSPACE-complete." Combined complexity refers to both the sentence and the graph being part of the input of the model checking problem. Two principal ways of dealing with the hardness of model checking problems are restrictions of the logics and restrictions of the classes of input graphs. In this survey, we shall only consider restrictions of the classes of input graphs. As for restrictions of the logics, let me just mention that the model checking problem remains NP-hard even for the fragment of first-order logic whose formulas are the *positive primitive formulas*, that is, existentially quantified conjunctions of atomic formulas. On the other hand, the model checking problem is in polynomial time for the *bounded variable fragments* of first-order logic [80].

Unfortunately, restricting the class of input graphs does not seem to improve the complexity, because the hardness result in Fact 2.6 can be strengthened: Let G be any graph with at least two vertices. Then

it is PSPACE-hard to decide whether a given FO-sentence φ holds in the fixed graph G. Of course this implies the corresponding hardness result for MSO. Hence not only the combined complexity, but also the *expression complexity* of FO and MSO is PSPACE-complete. Expression complexity refers to the problem of deciding whether a given sentence holds in a fixed graph. The reason for the hardness result is that in graphs with at least two vertices we can take atoms of the form $x = y$ to represent Boolean variables and use this to reduce the PSPACE-complete satisfiability problem for *quantified Boolean formulas* to the model checking problem. Let us explicitly state the following consequence of this hardness result, where we call a class of graphs *nontrivial* if it contains at least one graph with at least two vertices.

Fact 2.7. *For every nontrivial class $\mathscr{C}$ of graphs, the problems* $\mathrm{MC}(\mathrm{FO}, \mathscr{C})$ *and* $\mathrm{MC}(\mathrm{MSO}, \mathscr{C})$ *are* PSPACE-*hard.*

So what can we possibly gain by restricting the class of input graphs of our model checking problems? As there are no polynomial time algorithms (unless PTIME = PSPACE) even for very simple classes $\mathscr{C}$ of input graphs, we have to relax our notion of "tractability". A drastic way of doing this is to consider *data complexity* instead of combined complexity, that is, consider the complexity of evaluating a fixed sentence of the logic in a given graph. The following fact implies that the data complexity of FO is in PTIME:

Fact 2.8. *There is an algorithm that solves* $\mathrm{MC}(\mathrm{FO})$ *in time* $O(k^2 \cdot n^k)$, *where n denotes the order of the input graph G and k the length of the input sentence φ.*

Even though FO and MSO have the same combined complexity and the same expression complexity, the following example shows that the two logics differ in their data complexity:

Example 2.9. It is easy to see that there is an MSO-formula *3-col* saying that a graph is 3-colourable. As the 3-colourability problem is NP-complete, this shows that the data complexity of MSO is NP-hard. ⌋

There are, however, nontrivial classes $\mathscr{C}$ of graphs such that the data complexity of MSO restricted to $\mathscr{C}$ is in PTIME. As we shall see later, an example of such a class is the class of all trees. Thus things are starting to get interesting.

Still, while we have seen that polynomial combined complexity is too restrictive, polynomial data complexity may be too liberal as a notion of tractability. Recall from the introduction that this survey is about *algorithmic meta theorems*, that is, uniform tractability results for classes of algorithmic problems defined in terms of logic. Fact 2.8 implies such a meta theorem: *Every graph property definable in first-order logic can be decided in polynomial time.* A serious draw back of this result is that it does not bound the degrees of the polynomial running times of algorithms deciding first-order properties. An important justification for PTIME being a reasonable mathematical model of the class of "tractable" (that is, efficiently solvable) problems is that most problems solvable in polynomial time are actually solvable by algorithms whose running time is bounded by polynomials of low degree, usually not more than three. However, this is not the case for parameterized families of polynomial time definable problems such as the family of first-order definable graph properties, for which the degree of the polynomials is unbounded. Or more plainly, even for a property that is defined by a fairly short first-order sentence, say, of length $k = 10$, an algorithm deciding this property in time $O(n^{10})$ hardly qualifies as efficient. A much more useful meta theorem would state that first-order definable graph properties can be decided "uniformly" in polynomial time, that is, in time bounded by polynomials of a fixed degree. Unfortunately, such a theorem does not seem to hold, at least not for first-order definable properties of the class of all graphs.

The appropriate framework for studying such questions is that of *parameterized complexity theory* [28, 40, 61]. A *parameterized problem* is a pair (P, κ), where P is a decision problem in the usual sense and κ is a polynomial time computable mapping that associates a natural number, called the *parameter*, with each instance of P.[1]

Here we are mainly interested in model checking problems parameterized by the length of the input formula. For a logic L and a class $\mathscr{C}$ of graphs, we let:

---

[1] At some places in this paper (the first time in Remark 3.19) we are dealing with "parameterized problems" where the parameterization is not polynomial time computable. Whenever this appears here, the parameterization is computable by an fpt algorithm (see below), and this is good enough for our purposes. The same issue is also discussed in Section 11.4 of [40].

> p-MC(L,$\mathscr{C}$)
>   *Instance:* A sentence φ of L and a graph $G \in \mathscr{C}$.
> *Parameter:* $|\varphi|$.
>   *Problem:* Decide if $G \models \varphi$.

A parameterized problem (P, κ) is *fixed-parameter tractable* if there is an algorithm deciding whether an instance x is in P in time

$$f(\kappa(x)) \cdot |x|^c, \tag{2.1}$$

for some computable function f and some constant c. We call an algorithm that achieves such a running time an *fpt algorithm*. Slightly imprecisely, we call f the *parameter dependence* of the algorithm and c the *exponent*. An fpt algorithm with exponent 1 is called a *linear fpt algorithm*. Similarly, fpt algorithms with exponents 2 and 3 are called *quadratic* and *cubic*. FPT denotes the class of all parameterized problems that are fixed-parameter tractable.

Hence a parameterized model checking problem is fixed-parameter tractable if and only if it is "uniformly" in polynomial time, in the sense discussed above. (By requiring the function f bounding the running time to be computable, we impose a slightly stronger uniformity condition than above. This is inessential, but technically convenient.)

Parameterized complexity theory is mainly concerned with the distinction between running times like $O(2^k \cdot n)$ (fpt) and $O(n^k)$ (not fpt). Running times of the latter type yield the parameterized complexity class XP. Intuitively, a problem is in XP if it can be solved by an algorithm whose running time is polynomial for fixed parameter values. Formally, XP is the class of all parameterized problems that can be decided in time

$$O(|x|^{f(\kappa(x))}),$$

for some computable function f. Hence essentially, the parameterized model checking problem for a logic is in XP if and only if the data complexity of the logic is polynomial time. The class XP strictly contains FPT; this is an easy consequence of the time hierarchy theorem.

There is an appropriate notion of *fpt reduction* and a wide range of parameterized complexity classes between FPT and XP.

Example 2.10. A *clique* in a graph is the vertex set of a complete subgraph. The *parameterized clique problem* is defined as follows:

> p-CLIQUE
>   *Instance:* A graph G and a natural number k.
> *Parameter:* k.
>   *Problem:* Decide if G has a clique of size k.

It is easy to see that p-CLIQUE $\in$ XP. It can be proved that p-CLIQUE is complete for the parameterized complexity class W[1] under fpt reductions [27]. ⌐

Example 2.11. The *parameterized dominating set problem* is defined as follows:

> p-DOMINATING-SET
>   *Instance:* A graph G and a natural number k.
> *Parameter:* k.
>   *Problem:* Decide if G has a dominating set of size k.

It is easy to see that p-DOMINATING-SET $\in$ XP. It can be proved that p-DOMINATING-SET is complete for the parameterized complexity class W[2] under fpt reductions [26]. ⌐

The parameterized complexity classes W[1] and W[2] form the first two levels of the so-called *W-hierarchy* of classes between FPT and XP. Yet another parameterized complexity class, located between the W-hierarchy and XP, is called AW[∗]. Thus we have

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \text{W[3]} \subseteq \cdots \subseteq \text{AW[∗]} \subseteq \text{XP}.$$

It is conjectured that all containments between the classes are strict.

Fact 2.12 (Downey, Fellows, and Taylor [29]). *p-MC(FO) is AW[∗]-complete under fpt reductions.*

Observe that by Example 2.9, *p*-MC(MSO) is not even in XP unless PTIME = NP.

This concludes our brief introduction to parameterized complexity theory. For proofs of all results mentioned in this section, I refer the reader to [40].

3 Monadic second-order logic on tree-like classes of graphs

The model checking problem for monadic second-order logic turns out to be tractable on trees and graph classes that are sufficiently similar to trees. A well-known measure for the similarity of a graph with a tree is *tree width*. In this article, however, we will work with *branch width* instead. The tree width and branch width of a graph are the same up to a factor of $3/2$, so the results are essentially the same. Some of the results, including Courcelle's theorem, may sound unfamiliar this way, but the reader can substitute "tree" for "branch" almost everywhere, and the results will remain true (up to constant factors, which we usually disregard anyway). Using branch width instead of tree width may make this article a bit more interesting for those who do not want to read the definition of tree width for the 100th time. However, the main reason for working with branch width is that it combines nicely with the other graph invariant that we shall study in this section, *rank width*. Indeed, both branch width and rank width of a graph are instances of the same abstract notion of branch width of a set function.

3.1 Trees

Let $\mathscr{T}$ denote the class of all trees. Recall that then $\mathscr{T}_{lb}$ denotes the class of labelled trees.

Theorem 3.1 (Folklore). *p-MC(MSO, $\mathscr{T}_{lb}$) is solvable by a linear fpt algorithm.*

We sketch two proofs of this theorem. Even though one may view them as "essentially the same", the first is more natural from an algorithmic point of view, while the second will be easier to generalise later.

First proof sketch. Using a standard encoding of arbitrary trees in binary trees via the "first-child/next-sibling" representation, we can reduce the model checking problem for monadic second-order logic on arbitrary labelled trees to the model checking problem for monadic second-order logic on labelled binary trees. By a well-known theorem due to Thatcher and Wright [78], we can effectively associate a (deterministic) bottom-up tree automaton A_φ with every MSO-sentence φ over binary trees such that a binary tree T satisfies φ if and only if the automaton A_φ accepts T. By simulating the run of A_φ on T, it can be checked in linear time whether A_φ accepts a tree T. □

Second proof sketch. Again, we first reduce the model checking problem to binary trees. Let T be a labelled binary tree, and let φ be a monadic second-order sentence, say, of quantifier rank q. For every $t \in V(T)$, let T_t be the subtree of T rooted in t. Starting from the leaves, our algorithm computes $\text{tp}_q^{\text{MSO}}(T_t, t)$ for every $t \in T$, using Lemma 2.3. Then it decides if $\varphi \in \text{tp}_q^{\text{MSO}}(T, r)$ for the root r of T. □

The fpt algorithms described in the two proofs of Theorem 3.1 are linear in the size of the input trees. Clearly, this is optimal in terms of n (up to a constant factor). But what about the parameter dependence, that is, the function f in an fpt running time $f(k) \cdot n$? Recall that a function $f : \mathbb{N}^n \to \mathbb{N}$ is *elementary* if it can be formed from the successor function, addition, subtraction, and multiplication using composition, projections, bounded addition of the form $\sum_{\ell \le m} g(n_1, \ldots, n_k, \ell)$, and bounded multiplication of the form $\prod_{\ell \le m} g(n_1, \ldots, n_k, \ell)$. Let $\exp^{(h)}$ denote the h-fold exponentiation defined by $\exp^{(0)}(n) = n$ and $\exp^{(h)}(n) = 2^{\exp^{(h-1)}(n)}$ for all $n, h \in \mathbb{N}$. It is easy to see that $\exp^{(h)}$ is elementary for all $h \ge 0$ and that if a function $f : \mathbb{N} \to \mathbb{N}$ is elementary then there is an $h \ge 0$ such that $f(n) \le \exp^{(h)}(n)$ for all $n \in \mathbb{N}$. It is well known that there is no elementary function f such that the number of states of the smallest automaton A_φ equivalent to an MSO-formula φ of length k is at most $f(k)$. It follows that the parameter dependence of our automata based fpt algorithm for *p*-MC(MSO, $\mathscr{T}$) is non-elementary. Similarly, the number of monadic second-order q-types is nonelementary in terms of q, and hence the type based fpt algorithm also has a

nonelementary parameter dependence. But this does not rule out the existence of other fpt algorithms with a better parameter dependence. The following theorem shows that, under reasonable complexity theoretic assumptions, no such algorithms exist, not even for first-order model checking:

Theorem 3.2 (Frick and Grohe [44]).

(1) *Unless* PTIME $=$ NP*, there is no fpt algorithm for* $p\text{-MC}(\text{MSO}, \mathscr{T})$ *with an elementary parameter dependence.*

(2) *Unless* FPT $=$ W[1]*, there is no fpt algorithm for* $p\text{-MC}(\text{FO}, \mathscr{T})$ *with an elementary parameter dependence.*

As almost all classes $\mathscr{C}$ of graphs we shall consider in the following contain the class $\mathscr{T}$ of trees, we have corresponding lower bounds for the model checking problems on these classes $\mathscr{C}$. The only exception are classes of graphs of bounded degree, but even for such classes, we have a triply exponential lower bound [44] (cf. Remark 4.12).

3.2 Branch decompositions

We first introduce branch decompositions in an abstract setting and then specialise them to graphs in two different ways.

3.2.1 Abstract branch decompositions

Let A be a nonempty finite set and $\kappa : 2^A \to \mathbb{R}$. In this context, the function κ is often called a *connectivity function*. A *branch decomposition* of (A, κ) is a pair (T, β) consisting of a binary tree T and a bijection $\beta : L(T) \to A$. (Recall that $L(T)$ denotes the set of leaves of a tree T.) We inductively define a mapping $\widetilde{\beta} : V(T) \to 2^A$ by letting

$$\widetilde{\beta}(t) = \begin{cases} \{\beta(t)\} & \text{if } t \text{ is a leaf,} \\ \widetilde{\beta}(t_1) \cup \widetilde{\beta}(t_2) & \text{if } t \text{ is an inner node with children } t_1, t_2. \end{cases}$$

The *width* of the branch decomposition (T, κ) is defined to be the number

$$\text{width}(T, \kappa) = \max\left\{ \kappa(\widetilde{\beta}(t)) \mid t \in V(T) \right\},$$

and the *branch width* of (A, κ), denoted by $\text{bw}(A, \kappa)$, is defined to be the minimum of the widths of all branch decompositions of (A, κ). We extend the definition of branch width to empty ground sets A by letting $\text{bw}(\emptyset, \kappa) = \kappa(\emptyset)$ for all $\kappa : \{\emptyset\} \to \mathbb{R}$. Note that $(\emptyset, \kappa)$ does not have a branch decomposition, because the empty graph, not being connected, is not a tree.

Usually, the connectivity functions κ considered for branch decompositions are integer-valued, symmetric, and submodular. A function $\kappa : 2^A \to \mathbb{R}$ is *symmetric* if $\kappa(B) = \kappa(A \setminus B)$ for all $B \subseteq A$, and it is *submodular* if

$$\kappa(B) + \kappa(C) \geq \kappa(B \cup C) + \kappa(B \cap C) \tag{3.1}$$

for all $B, C \subseteq A$.

Example 3.3. Let $A \subseteq \mathbb{R}^n$ be finite. For every $B \subseteq A$, let $r(B)$ be the dimension of the linear subspace of $\mathbb{R}^n$ generated by B, or equivalently, the rank of the matrix with column vectors B (defined to be 0 if $B = \emptyset$). Define $\kappa_{\text{lin}} : 2^A \to \mathbb{Z}$ by

$$\kappa_{\text{lin}}(B) = r(B) + r(A \setminus B) - r(A).$$

κ_{lin} measures the dimension of the intersection of the subspace generated by B and the subspace generated by $A \setminus B$. It is easy to see that κ_{lin} is symmetric and submodular.

For example, let

$$A = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\} \subseteq \mathbb{R}^4.$$
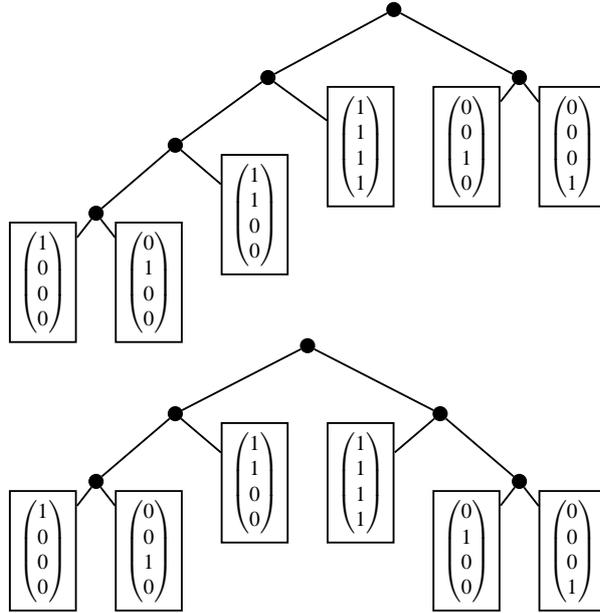
Figure 3.1. Two branch decompositions of (A, κ_{lin}) from Example 3.3

Figure 3.1 shows two branch decompositions of (A, κ_{lin}). I leave it as an exercise for the reader to verify that the first decomposition has width 1 and the second has width 2. Observe that $\text{bw}(A, \kappa_{\text{lin}}) = 1$, because every branch decomposition (T, β) of (A, κ_{lin}) has a leaf $t \in L(T)$ with $\beta(t) = (1,1,1,1)^T$, and we have $\kappa_{\text{lin}}(\{(1,1,1,1)^T\}) = 1$. ⌟

Example 3.4. Again, let $A \subseteq \mathbb{R}^n$. Now, for $B \subseteq A$ let $d(B)$ be the dimension of the affine subspace of $\mathbb{R}^n$ spanned by B (defined to be -1 if $B = \emptyset$), and let

$$\kappa_{\text{aff}}(B) = d(B) + d(A \setminus B) - d(A).$$

It is not hard to prove that κ_{aff} is also symmetric and submodular.

Figure 3.2 shows an example of a set $A = \{a, b, c, d, e, f, g, h\} \subseteq \mathbb{R}^2$ and a branch decomposition of (A, κ_{aff}) of width 1. ⌟

Example 3.5. The previous two examples have a common generalisation, which is known as the branch width of *matroids*.[2] Let M be a matroid with base set A and rank function r_M. Then the function $\kappa : 2^A \to \mathbb{Z}$ defined by

$$\kappa_M(B) = r_M(B) + r_M(A \setminus B) - r_M(A)$$

is known as the *connectivity function* of the matroid.[3] Obviously, κ_M is symmetric, and as the rank function r_M is submodular, κ_M is also submodular. ⌟

Before we return to graphs, let us state a very general algorithmic result, which shows that approximately optimal branch decompositions can be computed by an fpt algorithm. The proof of this theorem is beyond the scope of this survey. It is based on a deep algorithm for minimizing submodular functions due to Iwata, Fleischer, and Fujishige [52].

When talking about algorithms for branch decompositions, we have to think about how the input of these algorithms is specified. Let $\mathscr{A}$ be a class of pairs (A, κ), where $\kappa : 2^A \to \mathbb{Z}$ is symmetric and submodular and takes only nonnegative values. We call $\mathscr{A}$ a *tractable class of connectivity functions*, if we have a

---

[2]Readers who do not know anything about matroids should not worry. This example is the only place in this survey where they appear.

[3]Often, the connectivity function is defined by $\kappa_M(B) = r_M(B) + r_M(A \setminus B) - r_M(A) + 1$, but this difference is inessential here.
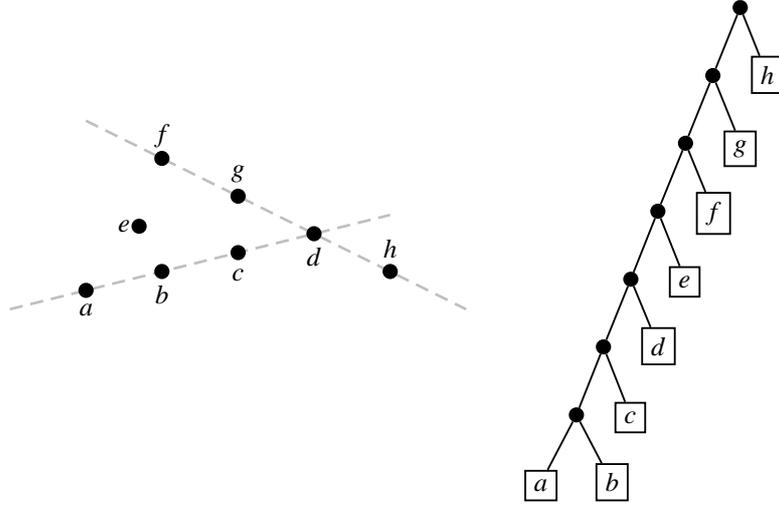
Figure 3.2. A set of A of eight points in the plane and a branch decomposition of (A, κ_{aff}) of width 1

representation of the pairs $(A, \kappa) \in \mathscr{A}$ such that, given the representation of (A, κ), the ground set A can be computed in polynomial time, and for every $B \subseteq A$, the value $\kappa(B)$ can be computed in polynomial time.

For example, if $\mathscr{A}$ is the class of pairs (A, κ_{lin}), where A is a finite set of vectors over some finite field or the field of rationals and κ_{lin} is the linear connectivity function, then we can represent a pair (A, κ_{lin}) simply by a matrix whose columns are the vectors in A. For the graph based examples that we shall describe next, the underlying graph is always used as a representation.

Theorem 3.6 (Oum and Seymour [63]). *Let $\mathscr{A}$ be a tractable class of connectivity functions. Then there is an fpt algorithm that, given $(A, \kappa) \in \mathscr{A}$ and a parameter $k \in \mathbb{N}$, computes a branch decomposition of (A, κ) of width at most $3k$ if $\mathrm{bw}(A, \kappa) \leq k$. If $\mathrm{bw}(A, \kappa) > k$, the algorithm may still compute a branch decomposition of (A, κ) of width at most $3k$, or it may simply halt without an output.*[4]

3.2.2 Branch decompositions of graphs

Let $G = (V, E)$ be a graph. For a set $F \subseteq E$, we define the *boundary* of F to be the set ∂F of all vertices of G incident both with an edge in F and with an edge in $E \setminus F$. We define a function $\kappa_G : 2^E \to \mathbb{Z}$ by letting $\kappa_G(F) = |\partial F|$ for every $F \subseteq E$. It is not hard to verify that κ_G is symmetric and submodular. A *branch decomposition* of G is a branch decomposition of (E, κ_G), and the *branch width* $\mathrm{bw}(G)$ of G is the branch width of (E, κ_G).

Example 3.7. Figure 3.3 shows an example of a graph and branch decomposition of this graph of width 5.
⌟

Example 3.8 ([70]). For every $n \geq 3$, the complete graph K_n on n-vertices has branch width $\lceil 2n/3 \rceil$.

We omit the proof of the lower bound. For the upper bound, we partition the vertex set of K_n into three parts V_1, V_2, V_3 of size $\lceil n/3 \rceil$ or $\lfloor n/3 \rfloor$, and we partition the edge set into three sets E_{12}, E_{23}, E_{13} such that edges in E_{ij} are only incident with vertices in $V_i \cup V_j$. Then we take arbitrary branch decompositions of the three subgraphs $G_{ij} = (V_i \cup V_j, E_{ij})$ and join them together as indicated in Figure 3.4. ⌟

Note that the construction of the previous example actually shows that every n-vertex graph has branch width at most $\lceil 2n/3 \rceil$.

Example 3.9 ([70]). A graph has branch width 0 if and only if it has maximum degree at most 1. A graph has branch width 1 if and only if it has at least one vertex of degree greater than 1, and every connected component has at most one vertex of degree greater than 1. Trees and cycles have branch width at most 2.

---

[4]An fpt algorithm of this type is known as an *fpt approximation algorithm* [7].

Figure 3.3. A graph with a branch decomposition of width 5. The numbers at the nodes indicate the size of the boundary of the edges in the subtree below that node.

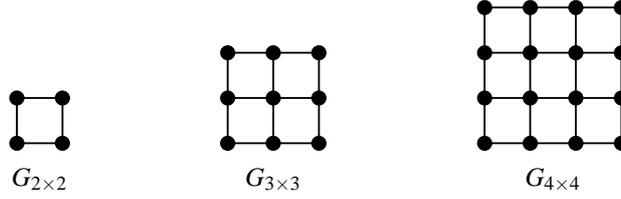**Figure 3.4.** A branch decomposition of a clique (see Example 3.8)

Figure 3.5. The $(n \times n)$-grids for $n = 2, 3, 4$

Let me suggest it as an exercise for the reader to prove these simple facts. ⌐

Example 3.10 ([70]). For all $n \geq 2$, the $n \times n$-grid

$$G_{n \times n} = \Big([n] \times [n], \big\{\{(i_1, j_1), (i_2, j_2)\} \mid |i_1 - i_2| + |j_1 - j_2| = 1\big\}\Big)$$

(cf. Figure 3.5) has branch width n. ⌐

Branch width is closely related to the more familiar *tree width*. In fact, it is not very hard to prove the following inequalities for all graphs G [70]:

$$\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq \max\big\{(3/2) \cdot \mathrm{bw}(G), 2\big\}, \tag{3.2}$$

where $\mathrm{tw}(G)$ denotes the tree width of G.

As the connectivity functions κ_G are symmetric and submodular, approximately optimal branch decompositions can be computed by the general purpose algorithm of Theorem 3.6. However, for the special case of branch decompositions of graphs, better algorithms are known:

Theorem 3.11 (Bodlaender and Thilikos [6]). *There is an algorithm that, given a graph G and a $k \in \mathbb{N}$, decides if $\mathrm{bw}(G) \leq k$ and computes a branch decomposition of G of width at most k if this is the case in time*

$$f(k) \cdot n,$$

where $n = |V(G)|$, for some computable function f.

3.2.3 Rank decompositions of graphs

Whereas branch width is based on decompositions of the edge set of a graph, for rank width we decompose its vertex set. For a graph $G = (V, E)$ and subsets $U, W \subseteq V$ of its vertex set, we let $M_G(U, W)$ be the $|U| \times |W|$-matrix with entries m_{uw} for $u \in U, w \in W$, where

$$m_{uw} = \begin{cases} 1 & \text{if } \{u, w\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Hence $M_G(V, V)$ is just the adjacency matrix of G. We view $M_G(U, W)$ as a matrix over the field $\mathrm{GF}(2)$ and let $\mathrm{rk}(M_G(U, W))$ be its row rank over $\mathrm{GF}(2)$. Now we define a connectivity function $\rho_G : 2^V \to \mathbb{N}$ by

$$\rho_G(U) = \mathrm{rk}\big(M_G(U, V \setminus U)\big)$$

for all $U \subseteq V$. Since the row rank and column rank of a matrix coincide, the function ρ_G is symmetric, and it is not hard to prove that it is submodular. A *rank decomposition* of G is a branch decomposition of (V, ρ_G), and the *rank width* $\mathrm{rw}(G)$ of G is the rank width of (V, ρ_G).

Example 3.12. Figure 3.6 shows an example of a graph and a rank decomposition of this graph of width 1. ⌐

It is easy to prove that rank width can be bounded in terms of branch width. The following theorem, which gives a tight bound, is not so obvious:
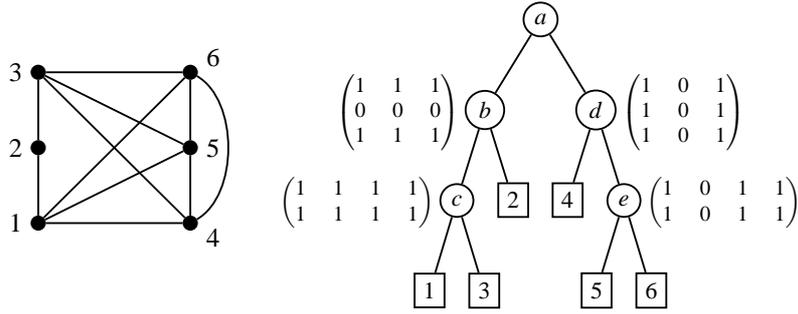
Figure 3.6. A graph with a rank decomposition of width 1. For later reference, we have named the nodes of the tree

Theorem 3.13 (Oum [62]). *For every graph G it holds that* $\mathrm{rw}(G) \le \max\{1, \mathrm{bw}(G)\}$.

The following example shows that the rank width of a graph can be substantially smaller than the branch width, and that it can also be the same.

Example 3.14. It is easy to see that every rank decomposition of a complete graph has width 1. Combined with Example 3.8, this shows that the branch width and rank width of a graph can differ by a factor $\Omega(n)$, where n denotes the number of vertices.

Let $I(K_n)$ be the graph obtained from the complete n-vertex graph K_n by subdividing all edges once, that is, by replacing every edge by a path of length 2. $I(K_n)$ is the *incidence graph* of K_n. Then if $n \ge 3$ and $n \equiv 0, 1 \mod 3$ we have $\mathrm{rw}(I(K_n)) = \mathrm{bw}(I(K_n)) = \lceil (2/3) \cdot n \rceil$ [62]. ⌟

Example 3.15. It can be shown that the rank width of an $(n \times n)$-grid is at least $\lceil n/2 - 2 \rceil$ (follows from [62]). Hence grids have both large branch width and large rank width. ⌟

As for the branch width of graphs, there is an algorithm for computing rank width that is more efficient than the general purpose algorithm of Theorem 3.6.

Theorem 3.16 (Hlineny and Oum [33]). *There is an algorithm that, given a graph G and a $k \in \mathbb{N}$, decides if $\mathrm{rw}(G) \le k$ and computes a rank decomposition of G of width at most k if this is the case in time*

$$f(k) \cdot n^3,$$

where $n = |V(G)|$, for some computable function f.

Rank width is related to the graph invariant *clique width* [17], which is defined in terms of a graph algebra: The clique width $\mathrm{cw}(G)$ of a graph G is the least number of constant symbols required in a term in this algebra describing the graph G. Oum and Seymour [63] proved that for every graph G it holds that

$$\mathrm{rw}(G) \le \mathrm{cw}(G) \le 2^{rw(G)+1} - 1.$$

In particular, this implies that a class of graphs has bounded rank width if and only if it has bounded clique width.

3.3 Courcelle's Theorems

For every $k \ge 1$, let $\mathscr{B}_k$ be the class of all graphs of branch width at most k and $\mathscr{R}_k$ the class of all graphs of rank width at most k. The following theorem is usually formulated in terms of tree width, but by (3.2) the following "branch width version" is equivalent.

Courcelle's Theorem ([9]). *For every k, the problem p-$\mathrm{MC}(\mathrm{MSO}, \mathscr{B}_k)$ is solvable by a linear fpt algorithm.*

As for Theorem 3.1, we sketch two proofs. The first is a reduction to Theorem 3.1, whereas the second is a generalisation of the second proof of Theorem 3.1.

First proof sketch. Let us fix $k \geq 1$. We reduce the model checking problem on the class $\mathscr{B}_k$ to that on labelled trees and then apply Theorem 3.1. We associate with each graph $G \in \mathscr{B}_k$ a labelled tree T^+ and with each MSO-sentence φ over graphs a sentence φ^+ over labelled trees such that $G \models \varphi \iff T^+ \models \varphi^+$. We will do this in such a way that T^+ is computable from G in linear time and that φ^+ is computable from φ. Then our model checking algorithm proceeds as follows: Given $G \in \mathscr{B}_k$ and $\varphi \in \text{MSO}$, it computes T^+ and φ^+ and then tests if T^+ satisfies φ^+ using the algorithm of Theorem 3.1.

The mapping $G \mapsto T^+$ will not be canonical, i.e., isomorphic graphs G will not necessarily yield isomorphic trees T^+. The tree T^+ will depend on the specific representation of the input graph G and on the algorithm we use to compute a branch decomposition of this input graph. Note that this does not affect the correctness of our algorithm.

We construct T^+ from G as follows: Without loss of generality we assume that G has no isolated vertices. We first compute a branch decomposition (T, β) of G of width at most k, which can be done in linear time by Theorem 3.11. Then we define a labelling of T that allows us to reconstruct G from the labelled tree T^+ *within MSO*. Formally, we define the labelling in such a way that G is *MSO-interpretable* in T^+. Then we can construct φ^+ from φ using the method of syntactic interpretations (see [32, 12]).

We assume that T is an ordered binary tree, that is, each inner node has a left and a right child. Recall that, for a node t of T, $\tilde{\beta}(t)$ is the set of all edges e of G such that $e = \beta(u)$ for some leaf u of T that appears in the subtree rooted at t. Let $B_t = \partial \tilde{\beta}(t)$ be the boundary of $\tilde{\beta}(t)$, that is, the set of all vertices incident with an edge in $\tilde{\beta}(t)$ and with an edge in $E(G) \setminus \tilde{\beta}(t)$. Since the width of (T, β) is at most k we have $|B_t| \leq k$ for all nodes t. The labelling of the tree T^+ encodes for every inner node t with left child t_1 and right child t_2 how B_t intersects the sets B_{t_1} and B_{t_2}. We assume some linear order of the vertices of G. Then there will be labels P_{1ij}, for $i, j \in [k]$, indicating that the ith vertex in B_{t_1} is equal to the jth vertex in B_t, and similarly labels P_{2ij} for t_2. Note that $B_t \subseteq B_{t_1} \cup B_{t_2}$, so these labels "determine" B_t. We do not label the leaves.

For each leaf t, the set B_t consists of the two endpoints of the edge $\beta(t)$ (unless one or both endpoints have degree 1). It is easy to write down four MSO-sentences $eq_{ij}(x, y)$, for $i, j \in \{0, 1\}$, such that for all leaves u, t of T we have $T^+ \models eq_{ij}(u, v)$ if and only if the ith vertex in B_u is equal to the jth vertex in B_t. Recalling our assumption that G has no isolated vertices, it is now easy to reconstruct G from T^+ within MSO. □

Second proof sketch. Let G be a graph of branch width k, and let φ be an MSO-sentence, say, of quantifier rank q. We compute a branch decomposition (T, β) of G of width k. We fix some linear order on the vertices of G. For every $t \in V(T)$ we let $\bar{b}_t$ be the ordered tuple of the elements of $\partial \tilde{\beta}(t)$. Recall that for a subset $B \subseteq E(G)$, by $G[\![B]\!]$ we denote the subgraph $(\bigcup B, B)$ generated by B.

Starting from the leaves we inductively compute $\text{tp}_q(G[\![\tilde{\beta}(t)]\!], \bar{b}_t)$ for all $t \in V(T)$, applying Lemma 2.3 at every node. For this to work, it is important that for all nodes t with children t_1 and t_2 it holds that

$$V\left(G[\![\tilde{\beta}(t_1)]\!] \cap G[\![\tilde{\beta}(t_2)]\!]\right) \subseteq \partial \tilde{\beta}(t_1) \cup \partial \tilde{\beta}(t_2)$$
$$\text{and } \partial \tilde{\beta}(t) \subseteq \partial \tilde{\beta}(t_1) \cup \partial \tilde{\beta}(t_2).$$

Finally, we check if $\varphi \in \text{tp}_q(G[\![\tilde{\beta}(r)]\!], \bar{b}_r)$ for the root r. (Note that $\bar{b}_r$ is actually the empty tuple, but this does not matter.) □

The following theorem was first proved by Courcelle [8, 11] in a version phrased in terms of certain graph grammars. Later, a version for clique width was proved by Courcelle, Makowsky, and Rotics [14], and finally the relation between clique width and rank width was established by Oum and Seymour [63].

Theorem 3.17 ([8, 11, 14, 63]). *For every k, p-$\text{MC}(\text{MSO}, \mathscr{R}_k)$ is solvable by a cubic fpt algorithm.*

Proof sketch. The proof follows the same strategy as the first proof of Courcelle's Theorem: We fix k. For every graph $G \in \mathscr{R}_k$ we construct a labelled tree T^* such that G can be reconstructed from T^* within MSO. Then using the method of syntactic interpretations, for every MSO-sentence φ over graphs we obtain an MSO-sentence φ^* over labelled trees such that $G \models \varphi \iff T^* \models \varphi^*$.

T^* is obtained by suitably labelling the tree T of a rank decomposition (T,β) of G of width k. The difficulty here is to encode G in a labelling of T that uses only finitely many labels. Let t be an inner node of T with children t_1 and t_2. For $i = 1,2$, let $U_i = \tilde{\beta}(t_i)$. Furthermore, let $U = U_1 \cup U_2$ and $W = V \setminus U$. Then $\tilde{\beta}(t) = U$, and the matrices at the nodes t_1, t_2, t can be written as

$$M(U_1, V \setminus U_1) = \big(M(U_1, U_2) \quad M(U_1, W)\big),$$
$$M(U_2, V \setminus U_2) = \big(M(U_2, U_1) \quad M(U_2, W)\big),$$
$$M(U, V \setminus U) = \begin{pmatrix} M(U_1, W) \\ M(U_2, W) \end{pmatrix}.$$

Note that $M(U_2, U_1)$ is the transpose of $M(U_1, U_2)$. (We omit the subscript $_G$ for the matrices $M_G(\cdot, \cdot)$.)

For every node $t \in V(T)$ we compute a set B_t of at most k vertices of G such that the rows corresponding to the vertices in B_t form a basis of the row space of the matrix $M(U, V \setminus U)$, where $U = \tilde{\beta}(t)$. We define a labelling of the (inner) nodes of T as follows: Let t be an inner node with children t_1 and t_2 and $U_1 = \tilde{\beta}(t_1)$, $U_2 = \tilde{\beta}(t_2)$, $U = U_1 \cup U_2 = \tilde{\beta}(t)$. Then at t the labelling encodes

- the matrix $M(B_{t_1}, B_{t_2})$,

- for $i = 1,2$ and each $v \in B_{t_i}$ a representation of the row of $M(U, V \setminus U)$ corresponding to v as a linear combination of vectors of the basis corresponding to B_t over the field GF(2).

Note that this amounts to at most $3k^2$ bits of information: The matrix requires at most k^2 bits, and a linear combination of k vectors over GF(2) requires k bits.

We now describe how the graph G can be reconstructed from the labelled tree T^*. The vertices of G correspond to the leaves of T^*. To find out whether there is an edge between a vertex v_1, say, with $v_1 = \beta(u_1)$ and a vertex v_2, say with $v_2 = \beta(u_2)$, we proceed as follows: Let t be the first common ancestor of u_1 and u_2, and let t_1 and t_2 be the children of t such that u_i is a descendant of t_i, for $i = 1,2$. Let $U_i = \tilde{\beta}(t_i)$ and $U = U_1 \cup U_2 = \tilde{\beta}(t)$. Then $v_i \in U_i$. Note that $B_{u_i} = \{v_i\}$, because the matrices at the leaves only have one row. Hence, using the labelling, we can recursively find a representation of the row of the matrix $M(U_i, V \setminus U_i)$ corresponding to v_i as a linear combination of the rows corresponding to B_{t_i}. Then we can use the matrix $M(B_{t_1}, B_{t_2})$, which is also part of the labelling, to compute the entry $m_{v_1 v_2}$ of the matrix $M(U_1, U_2)$, and this entry tells us whether there is an edge between v_1 and v_2. The following example illustrates this construction. $\square$

Example 3.18. Consider the graph G and branch decomposition displayed in Figure 3.6. We define the "bases" as follows:

t	1	2	3	4	5	6	a	b	c	d	e
B_t	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$	$\{6\}$	$\emptyset$	$\{1\}$	$\{1\}$	$\{4\}$	$\{5\}$

Then for example, at node b the following information is stored: The matrix

$$M(\{1\}, \{2\}) = (1),$$

and a representation of the rows $r_1 = (1\,1\,1)$ and $r_2 = (0\,0\,0)$ of the matrix $M(\{1,2,3\}, \{4,5,6\})$ in terms of the row r_1:

$$r_1 = 1 \cdot r_1, \quad r_2 = 0 \cdot r_1.$$

To determine whether there is an edge, say, between between $v_1 = 3$ and $v_2 = 5$ we take the least common ancestor of the two leaves, a with its two children b and d. The representation of row $r_3 = (1\,1\,1)$ of $M(\{1,2,3\}, \{4,5,6\})$ with respect to $B_b = \{1\}$ is $r_3 = 1 \cdot r_1$, and the representation of row $r_5 = (1\,0\,1)$ of $M(\{4,5,6\}, \{1,2,3\})$ with respect to $B_d = \{4\}$ is $r_5 = 1 \cdot r_4$. Hence $m_{35} = 1 \cdot 1 \cdot m_{14} = 1$, that is, there is an edge between 3 and 5. $\lrcorner$

It follows from Theorem 3.2 that the parameter dependence of the fpt algorithms in the previous two theorems has to be nonelementary.

We close this section with two remarks about strengthenings of the two theorems:

Remark 3.19. Our proofs yield stronger theorems than stated: Not only is the MSO model checking problem fixed-parameter tractable on every class of graphs whose branch width is bounded, but actually the following doubly parameterized model checking problem is fixed-parameter tractable:

> *Instance:* A sentence $\varphi \in$ MSO and a graph G.
> *Parameter:* $|\varphi| + \text{bw}(G)$.
> *Problem:* Decide if $G \models \varphi$.

The same is true for rank width. ⌟

Remark 3.20. It is easy to see that both theorems can be extended to labelled graphs.

Courcelle's Theorem even holds for a stronger monadic second order logic, denoted by MSO_2, that admits quantification not only over sets of vertices of a graph, but also over sets of edges. This stronger result can easily be derived from the (labelled) version of Courcelle's Theorem. Define the *incidence graph* $I(G)$ of a graph G to be the graph (V_I, E_I), where $V_I = V(G) \cup E(G)$ and $E_I = \{\{v, e\} \mid v \in e\}$. It is not hard to see that for every graph G of branch width at least 2 it holds that $\text{bw}(G) = \text{bw}(I(G))$. Furthermore, every MSO_2-formula over G can be translated to an MSO-formula over the labelled incidence graph $(I(G), P)$, where $P = E(G)$ (The labelling is not really needed, but convenient.) Hence it follows from Courcelle's Theorem that $p\text{-MC}(\text{MSO}_2, \mathscr{B}_k)$ has a linear fpt algorithm for every $k \geq 1$.

This does not work for rank width, because the rank width of the incidence graph can be much larger than that of the original graph. Surprisingly, the rank width of the incidence graph of a graph is closely related to the branch width of the original graph. Oum [62] proved that

$$\text{bw}(G) - 1 \leq \text{rw}(I(G)) \leq \text{bw}(G)$$

for every graph G with at least one vertex of degree 2. ⌟

4 First-order logic on locally tree-like classes of graphs

There is not much hope for extending the tractability of monadic second-order model checking to further natural classes of graphs such as planar graphs or graphs of bounded degree. Indeed, the MSO-definable 3-colourability problem is NP-complete even when restricted to planar graphs of degree 4. For first-order logic, however, the model checking problem is tractable on much larger classes of graphs. Seese [75] showed that first-order model checking admits a linear fpt algorithm on all classes of bounded degree. Later Frick and Grohe [43] proved the same for planar graphs, essentially by the general approach that we shall describe in this section. The crucial property of first-order logic that we exploit is its *locality*.

4.1 The Locality of First-Order Logic

Let $G = (V, E)$ be a graph. The *distance* $\text{dist}^G(v, w)$ between two vertices $v, w \in V$ is the length of the shortest path from v to w. For every $v \in V$ and $r \in \mathbb{N}$, the *r-neighbourhood* of v in G is the set

$$N_r^G(v) = \{w \in V \mid \text{dist}^G(v, w) \leq r\}$$

of all vertices of distance at most r from v. For a set $W \subseteq V$, we let $N_r^G(W) = \bigcup_{w \in W} N_r^G(w)$. We omit the superscript ^{G} if G is clear from the context. The *radius* of a connected graph G is the least r for which there is a vertex $v \in V(G)$ such that $V(G) \subseteq N_r(v)$. The radius of a disconnected graph is ∞.

Observe that distance is definable in first-order logic, that is, for every $r \geq 0$ there is a first-order formula $dist_{\leq r}(x, y)$ such that for all graphs G and $v, w \in V(G)$,

$$G \models dist_{\leq r}(v, w) \iff \text{dist}(v, w) \leq r.$$

In the following, we will write $dist(x, y) \leq r$ instead of $dist_{\leq r}(x, y)$ and $dist(x, y) > r$ instead of $\neg dist_{\leq r}(x, y)$.

A first-order formula $\varphi(x_1, \ldots, x_k)$ is *r-local* if for every graph G and all $v_1, \ldots, v_k \in V(G)$ it holds that

$$G \models \varphi(v_1, \ldots, v_k) \iff G[N_r(\{v_1, \ldots, v_k\})] \models \varphi(v_1, \ldots, v_k).$$

This means that it only depends on the r-neighbourhood of a vertex tuple whether an r-local formula holds at this tuple. A formula is *local* if it is r-local for some r.

A *basic local sentence* is a first-order sentence of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} dist(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{k} \varphi(x_i) \right),$$

where $\varphi(x)$ is r-local. In particular, for every local formula $\varphi(x)$ the sentence $\exists x \, \varphi(x)$ is a basic local sentence.

Gaifman's Locality Theorem ([45]). *Every first-order sentence is equivalent to a Boolean combination of basic local sentences.*

Furthermore, there is an algorithm that computes a Boolean combination of basic local sentences equivalent to a given first-order sentence.

We shall illustrate the following proof sketch in Example 4.1 below. To appreciate the cleverness of the proof, the reader may try to find a Boolean combination of basic local sentences equivalent to the simple sentence $\varphi = \exists x \exists y (\neg E(x,y) \wedge P(x) \wedge Q(y))$ considered in the example before reading the proof.

Proof sketch. The proof is by structural induction on first-order formulas. To enable this induction, we need to prove a stronger statement that also includes formulas with free variables. We say that a first-order formula is in *Gaifman normal form (GNF)* if it is a Boolean combination of basic local sentences and local formulas.

Claim: Every first-order formula is equivalent to a formula in GNF.

The claim is trivial for atomic formulas, because all atomic formulas are 0-local. It obviously extends to Boolean combinations of formulas. Universal quantification can be reduced to existential quantfication and negation. The only remaining case is that of existentially quantified formulas

$$\varphi(\bar{x}) = \exists y \, \psi(\bar{x}, y),$$

where $\psi(\bar{x}, y)$ is in GNF. We may assume that $\psi(\bar{x}, y)$ is of the form

$$\bigvee_{i=1}^{m} \left(\chi_i \wedge \xi_i(\bar{x}, y) \right),$$

where each χ_i is a Boolean combination of basic local sentences and each $\xi_i(\bar{x}, y)$ is local. Here we use the simple observation that a Boolean combination of local formulas is local. Then $\varphi(\bar{x})$ is equivalent to the formula

$$\bigvee_{i=1}^{m} \left(\chi_i \wedge \exists y \, \xi_i(\bar{x}, y) \right).$$

It remains to prove that each formula

$$\varphi'(\bar{x}) = \exists y \, \xi(\bar{x}, y),$$

where $\xi(\bar{x}, y)$ is local, is equivalent to a formula in GNF. Let $r \geq 0$ such that $\xi(\bar{x}, y)$ is r-local. We observe that $\varphi'(\bar{x})$ is equivalent to the formula

$$\exists y (dist(\bar{x}, y) \leq 2r + 1 \wedge \xi(\bar{x}, y)) \vee \exists y (dist(\bar{x}, y) > 2r + 1 \wedge \xi(\bar{x}, y)), \tag{4.1}$$

where $dist(\bar{x}, y) \leq 2r + 1$ abbreviates $\bigvee_i dist(x_i, y) \leq 2r + 1$. The first formula in the disjunction (4.1) is $(3r + 1)$-local. Hence we only need to consider the second, $\exists y (dist(\bar{x}, y) > 2r + 1 \wedge \xi(\bar{x}, y))$. Using Lemma 2.3 and the r-locality of $\xi(\bar{x}, y)$, it is not hard to see that this formula is equivalent to a Boolean combination of formulas of the form

$$\zeta(\bar{x}) \wedge \exists y (dist(\bar{x}, y) > 2r + 1 \wedge \eta(y)),$$

19

where $\zeta(\bar{x})$ and $\eta(y)$ are r-local. Let $r' = 2r + 1$. It remains to prove that

$$\varphi''(\bar{x}) = \exists y \big(dist(\bar{x}, y) > r' \wedge \eta(y) \big)$$

is equivalent to a formula in GNF. This is the core of the whole proof. Suppose that $\bar{x} = (x_1, \ldots, x_k)$. Let G be a graph and $\bar{v} = (v_1, \ldots, v_k) \in V(G)^k$. When does $G \models \varphi''(\bar{v})$ hold? Clearly, it holds if there are $w_1, \ldots, w_{k+1}$ of pairwise distance greater than $2r'$ such that $G \models \eta(w_j)$ for all j, because each r'-neighbourhood $N_{r'}(v_i)$ contains at most one w_j and hence there is at least one w_j of distance greater than r' from all the v_i. For $\ell \geq 1$, let

$$\theta_\ell = \exists y_1 \ldots \exists y_\ell \Big(\bigwedge_{1 \leq i < j \leq \ell} dist(y_i, y_j) > 2r' \wedge \eta(y_i) \Big).$$

Note that θ_ℓ is a basic local sentence. We have just seen that θ_{k+1} implies $\varphi''(\bar{x})$. But of course $\varphi''(\bar{x})$ may also hold if θ_{k+1} does not. Let us return to our graph G and the tuple $\bar{v} \in V(G)^k$. Let $\ell \geq 1$ be maximum such that $G \models \theta_\ell$ and suppose that $\ell \leq k$. In the following case distinction, we shall determine when $G \models \varphi''(\bar{v})$.

Case 1: There are no $w_1, \ldots, w_\ell \in N_{r'}(\{\bar{v}\})$ of pairwise distance greater than $2r'$ such that $G \models \eta(w_j)$ for all j.

As $G \models \theta_\ell$, this implies that there is at least one $w \notin N_{r'}(\{\bar{v}\})$ such that $G \models \eta(w)$. Hence $G \models \varphi''(\bar{v})$.

Case 2: There is a $w \in N_{3r'}(\bar{v})$ such that $w \notin N_{r'}(\bar{v})$ and $G \models \eta(w)$.

Then, trivially, $G \models \varphi''(\bar{v})$.

Case 3: Neither Case 1 nor Case 2, that is, there are $w_1, \ldots, w_\ell \in N_{r'}(\{\bar{v}\})$ of pairwise distance greater than $2r'$ such that $G \models \eta(w_j)$ for all j, and there is no $w \in N_{3r'}(\bar{v}) \setminus N_{r'}(\bar{v})$ such that $G \models \eta(w)$.

Then $G \not\models \varphi''(\bar{v})$. To see this, suppose for contradiction that there is a $w \in V(G)$ such that $w \notin N_{r'}(\{\bar{v}\})$ and $G \models \eta(w)$. Then $w \notin N_{3r'}(\{\bar{v}\})$ and therefore $dist(w_j, w) > 2r'$ for all $j \in [\ell]$. Thus $G \models \theta_{\ell+1}$, which contradicts the maximality of ℓ.

Hence $G \models \varphi''(\bar{v})$ if any only if we are in Case 1 or 2. Note that the conditions describing these cases can be defined by local formulas, say, $\gamma_{\ell,1}(\bar{x})$ and $\gamma_{\ell,2}(\bar{x})$. Thus if $G \models \theta_\ell \wedge \neg\theta_{\ell+1}$, then $G \models \varphi''(\bar{v})$ if and only if $G \models \gamma_{\ell,1}(\bar{v}) \vee \gamma_{\ell,2}(\bar{v})$.

Overall, $\varphi''(\bar{x})$ is equivalent to the formula

$$\theta_{k+1} \vee \bigvee_{\ell=1}^{k} \Big(\theta_\ell \wedge \neg\theta_{\ell+1} \wedge \big(\gamma_{\ell,1}(\bar{x}) \vee \gamma_{\ell,2}(\bar{x}) \big) \Big),$$

which is in GNF. It is not hard to show that our construction yields an algorithm that computes a formula in GNF equivalent to a given first-order formula. $\square$

Example 4.1. Let us follow the proof of Gaifman's theorem and construct a Boolean combination of basic local sentences equivalent to the sentence

$$\varphi = \exists x \exists y \big(\neg E(x, y) \wedge P(x) \wedge Q(y) \big),$$

which is a sentence over labelled graphs with labels P and Q.

The quantifier free formula $\varphi_0(x, y) = \big(\neg E(x, y) \wedge P(x) \wedge Q(y) \big)$ is 0-local. Hence we start the construction with the formula

$$\varphi_1(x) = \exists y \big(\neg E(x, y) \wedge P(x) \wedge Q(y) \big).$$

$\varphi_1(x)$ is equivalent to the formula

$$\varphi_1' = P(x) \wedge \exists y \big(\neg E(x, y) \wedge Q(y) \big).$$

Splitting $\exists y \big(\neg E(x, y) \wedge Q(y) \big)$ with respect to the distance between x and y as in (4.1) (with $r = 0$) and simplifying the resulting formula, we obtain

$$P(x) \wedge \Big(Q(x) \vee \exists y \big(dist(x, y) > 1 \wedge Q(y) \big) \Big).$$

20

It remains to consider the formula $\varphi_1''(x) = \exists y \big(dist(x,y) > 1 \wedge Q(y)\big)$. Following the proof of Gaifman's theorem (with $\varphi'' = \varphi_1''$, $\eta(y) = Q(y)$, $r = 0$, and $k = 1$), we obtain the following equivalent formula in GNF:

$$\varphi_1'''(x) = \theta_2 \vee \Big(\theta_1 \wedge \neg\theta_2 \wedge \big(\neg\exists y(dist(x,y) \leq 1 \wedge Q(y))$$
$$\vee \exists y(dist(x,y) \leq 3 \wedge dist(x,y) > 1 \wedge Q(y))\big)\Big)$$

where $\theta_1 = \exists y_1 Q(y_1)$ and $\theta_2 = \exists y_1 \exists y_2 \big(dist(y_1,y_2) > 2 \wedge Q(y_1) \wedge Q(y_2)\big)$. Hence $\varphi_1(x)$ is equivalent to the formula $P(x) \wedge \big(Q(x) \vee \varphi_1'''(x)\big)$. The step from $\varphi_1(x)$ to $\varphi = \exists x \varphi_1(x)$ is simple, because there are no free variables left. By transforming the formula $P(x) \wedge \big(Q(x) \vee \varphi_1'''(x)\big)$ into disjunctive normal form and pushing the existential quantifier inside, we obtain the formula:

$$\exists x \big(P(x) \wedge Q(x)\big)$$
$$\vee \big(\exists x\, P(x) \wedge \theta_2\big)$$
$$\vee \Big(\exists x \big(P(x) \wedge \neg\exists y(dist(x,y) \leq 1 \wedge Q(y))\big) \wedge \theta_1 \wedge \neg\theta_2\Big)$$
$$\vee \Big(\exists x \big(P(x) \wedge \exists y(dist(x,y) \leq 3 \wedge dist(x,y) > 1 \wedge Q(y))\big) \wedge \theta_1 \wedge \neg\theta_2\Big).$$

Observe that this is indeed a Boolean combination of basic local sentences equivalent to φ. A slightly simpler Boolean combination of basic local sentences equivalent to φ is constructed in Example 3 of [51] by a different technique. ⌟

It has recently been proved in [20] that the translation of a first-order sentence into a Boolean combination of basic local sentences may involve a nonelementary blow-up in the size of the sentence.

4.2   Localisations of graph invariants

Recall that $\mathscr{G}$ denotes the class of all graphs. For every graph invariant $f : \mathscr{G} \to \mathbb{N}$ we can define its *localisation* $\ell_f : \mathscr{G} \times \mathbb{N} \to \mathbb{N}$ by

$$\ell_f(G,r) = \max\Big\{ f\big(G[N_r(v)]\big) \mid v \in V(G)\Big\}.$$

Hence to compute $\ell_f(G,r)$, we apply f to every r-neighbourhood in G and then take the maximum. We say that a class $\mathscr{C}$ of graphs has *locally bounded* f if there is a computable[5] function $g : \mathbb{N} \to \mathbb{N}$ such that $\ell_f(G,r) \leq g(r)$ for all $G \in \mathscr{C}$ and all $r \in \mathbb{N}$.

Example 4.2. One of the simplest graph invariants is the order of a graph. Observe that a class of graphs has locally bounded order if and only if it has bounded degree.

Moreover, if a class $\mathscr{C}$ has bounded degree then it has locally bounded f for every computable graph invariant f. ⌟

In this section, we are mainly interested in the localisation of branch width. Maybe surprisingly, there are several natural classes of graphs of locally bounded branch width. We start with two trivial examples and then move on to more interesting ones:

Example 4.3. Every class of graphs of bounded branch width has locally bounded branch width. ⌟

Example 4.4. Every class of graphs of bounded degree has locally bounded branch width. This follows immediately from Example 4.2. ⌟

Example 4.5 ([68, 76]). The class of planar graphs has locally bounded branch width. More precisely, a planar graph of radius r has branch width at most $2r+1$.

Let me sketch the proof. Let G be a planar graph of radius r, and let v_0 be a vertex such that $V(G) \subseteq N_r(v_0)$. We show how to recursively partition the edge set of G in such a way that at each stage, the

(a) The graph is cut along *PQ*

(b) Part *B* is cut again along *R*

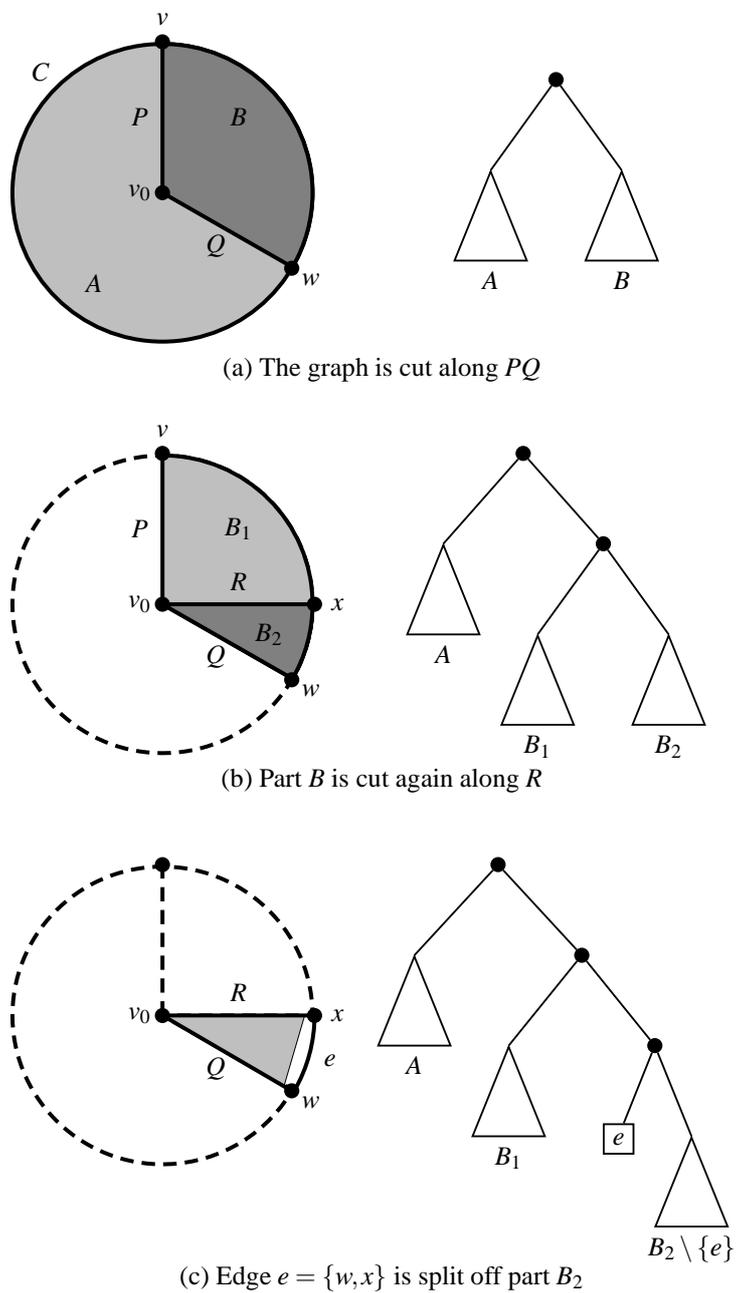(c) Edge $e = \{w, x\}$ is split off part B_2

Figure 4.1. Schematic branch decomposition of a planar graph

boundary of each part has cardinality at most $2r+1$. This will give us a branch decomposition of width at most $2r+1$.

Without loss of generality we may assume that G is 2-connected; if it is not, we first decompose it into its 2-connected blocks. Figure 4.1 illustrates the following steps. We fix a planar embedding of G, and let C be the exterior cycle. We pick two vertices v,w on C and shortest paths P,Q from v_0 to v,w, respectively. Then we cut along P and Q. This gives us a partition of $E(G)$ into two parts whose boundary is contained in $V(P \cup Q)$. We can add the edges in $E(P \cup Q)$ arbitrarily to either of the two parts. Now we consider each of the parts separately. The boundary cycle consists of P, Q, and a piece of the cycle C. If this piece of C is just one edge, we can split it off and then further decompose the rest. Otherwise, we pick a vertex x on the piece of C and a shortest path R from v_0 to x. We obtain two new parts with boundaries $V(P \cup R)$ and $V(Q \cup R)$. We partition these new parts recursively until they only consist of their boundaries, and then we partition the rest arbitrarily. Of course this proof sketch omits many details and special cases. For example, the vertex v_0 could be on the exterior cycle to begin with. I leave it to the reader to work out these details.

The branch decomposition in Figure 3.3 was obtained by this method. Note that the graph has radius 2, with centre v_0 being the vertex incident with the edges m and j. The initial paths P and Q have edge sets $E(P) = \{s,m\}$ and $E(Q) = \{j\}$. The right part consists of the edges $a,b,c,k,d,e,f,l,o,n,u,t,w,p,v,q$. The edges of $P \cup Q$ were added to the left part. In the next step, the right part was split along the path R with $E(R) = \{k,e\}$. The right part of this split consists of the edges f,l,o,n,u,t,w,p,v,q. The edge f immediately can be split off, and the new boundary cycle is w,q,l,k,m,s. The new splitting path consists of the edge o, et cetera. ⌐

Example 4.6 ([35]). The *genus* of a graph is the minimum genus of an orientable or nonorientable surface the graph can be embedded into. For every k, the class of all graphs of genus at most k has locally bounded branch width. Moreover, for every k the class of all graphs of *crossing number* at most k has locally bounded branch width. ⌐

In the next example, we shall construct an artificial class of graphs of locally bounded branch width. It serves as an illustration that the global structure of graphs of locally bounded branch width can be quite complicated. In particular, this example shows that there are classes of graphs of locally bounded branch width and of unbounded average degree. Recall that if a class $\mathscr{C}$ of graphs has unbounded average degree then the size of the graphs in $\mathscr{C}$ is superlinear in their order. The graph classes in all previous examples have bounded average degree and thus size linear in the order. For planar graphs and graphs of bounded genus, this follows from Euler's formula.

Example 4.7 ([43]). Recall that the *girth* of a graph is the length of its shortest cycle, and the *chromatic number* is the least number of colours needed to colour the graph in such a way that no two adjacent vertices receive the same colour. We shall use the well-known fact, due to Erdös [36], that for all $g,k \geq 1$ there exist graphs of girth greater than g and chromatic number greater than k. The proof of this fact (see [2]) shows that we can effectively construct such a graph $G_{g,k}$ for given g and k.

Then for every $k \geq 1$, every graph $G_{k,k}$ must have a subgraph H_k of minimum degree at least k; otherwise we could properly colour G with k colours by a straightforward greedy algorithm (see [25], Corollary 5.2.3). Let $H_k \subseteq G_{k,k}$ be such a subgraph. As a subgraph of $G_{k,k}$ the graph H_k still has girth greater than k.

Let $\mathscr{C} = \{H_k \mid k \geq 1\}$. Then $\mathscr{C}$ has unbounded minimum degree and hence unbounded average degree. Nevertheless, $\mathscr{C}$ has locally bounded branch width. To see this, simply observe that the r-neighbourhood of every vertex in a graph of girth greater than $2r+1$ is a tree. As the branch width of a tree is at most 2, for every graph $H \in \mathscr{C}$ and every $r \geq 1$ we have

$$\ell_{\text{bw}}(H,r) \leq \max\left(\left\{\text{bw}(H_k) \mid k \leq 2r+1\right\} \cup \{2\}\right). \qquad ⌐$$

4.3 Model checking algorithms

Theorem 4.8. *Let f be a graph invariant such that the following parameterization of the model checking problem for first-order logic is fixed-parameter tractable:*

---

[5]It would be more precise to call this notion "effectively locally bounded f", but this would make the terminology even more awkward.

> $p\text{-MC}(\text{FO}, f)$
>   *Instance:* A sentence $\varphi \in \text{FO}$ and a labelled graph G.
> *Parameter:* $|\varphi| + f(G)$.
>   *Problem:* Decide if $G \models \varphi$.

Then for every class $\mathscr{C}$ of graphs of locally bounded f, the problem $p\text{-MC}(\text{FO}, \mathscr{C})$ is fixed-parameter tractable.

The proof of the theorem relies on Gaifman's Locality Theorem and the following lemma:

Lemma 4.9 ([43]). *Let f and $\mathscr{C}$ be as in Theorem 4.8. Then the following problem is fixed-parameter tractable:*

>   *Instance:* A labelled graph $G = (V, E, P) \in \mathscr{C}_{lb}$ and $k, r \in \mathbb{N}$.
> *Parameter:* $k + r$.
>   *Problem:* Decide if there are vertices $v_1, \dots, v_k \in P$ such that $\text{dist}(v_i, v_j) > 2r$ for $1 \le i < j \le k$.

For simplicity, we only prove the lemma for graph invariants f that are *induced-subgraph-monotone*, that is, for all graphs G and induced subgraphs $H \subseteq G$ we have $f(H) \le f(G)$. Note that both branch width and rank width are induced-subgraph-monotone.

Proof sketch of Lemma 4.9. Given $G = (V, E, P)$ and $k, r \in \mathbb{N}$, we first compute a maximal (with respect to inclusion) set $S \subseteq P$ of vertices of pairwise distance greater than $2r$. If $|S| \ge k$, then we are done.

Otherwise, we know that $P \subseteq N_{2r}(S)$. Let H be the induced subgraph of G with vertex set $N_{3r}(S)$. As $|S| < k$, the radius of each connected component of H is at most $(3r+1) \cdot k$. Hence, because f is induced-subgraph-monotone,

$$f(H) \le \ell_f(G, (3r+1) \cdot k) \le g((3r+1) \cdot k),$$

where g is a function witnessing that $\mathscr{C}$ has locally bounded f.

Since $P \subseteq N_{2r}(S)$ and $V(H) = N_{3r}(S)$, for all vertices $v, w \in P$ it holds that $\text{dist}^G(v, w) > 2r$ if and only if $\text{dist}^H(v, w) > 2r$. Hence it remains to check whether H contains k vertices labelled P of pairwise distance greater than $2r$. This is equivalent to saying that H satisfies the first-order sentence

$$\exists x_1 \dots \exists x_k \left(\bigwedge_{1 \le i < j \le k} dist(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{k} P(x_i) \right).$$

We can use an fpt algorithm for $p\text{-MC}(\text{FO}, f)$ to check this. $\qquad\square$

Proof sketch of Theorem 4.8. Let $G = (V, E) \in \mathscr{C}$ and $\varphi \in \text{FO}$. We first transform φ into an equivalent Boolean combination of basic local sentences. Then we check separately for each basic local sentence in this Boolean combination whether it is satisfied by G and use the results to determine whether φ holds.

So let us consider a basic local sentence

$$\psi = \exists x_1 \dots \exists x_k \left(\bigwedge_{1 \le i < j \le k} dist(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{k} \chi(x_i) \right),$$

where $\chi(x)$ is r-local. For each vertex v of G we check whether $G[N_r(v)]$ satisfies $\chi(v)$ using an fpt algorithm for $p\text{-MC}(\text{FO}, f)$. We can do this within the desired time bounds because $f(G[N_r(v)]) \le \ell_f(G, r)$. If $G[N_r(v)]$ satisfies $\chi(v)$, we label v by P. To determine whether G satisfies ψ, we have to check whether the labelled graph (V, E, P) has k vertices in P of pairwise distance greater than $2r$. By Lemma 4.9, this can be done by an fpt algorithm. $\qquad\square$

Corollary 4.10 ([43]). *For every class $\mathscr{C}$ of graphs of locally bounded branch width, $p\text{-MC}(\text{FO}, \mathscr{C})$ is fixed-parameter tractable.*
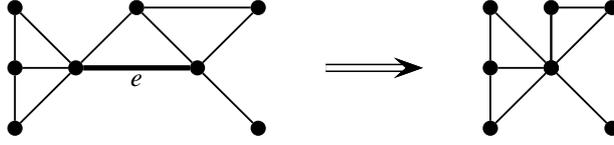
Figure 5.1. Contraction of edge e

Corollary 4.11. *For every class $\mathscr{C}$ of graphs of locally bounded rank width, $p\text{-MC}(\text{FO},\mathscr{C})$ is fixed-parameter tractable.*

Let me close this section with a few remarks on the running time of the model checking algorithms.

Remark 4.12. We first look at the exponent of the fpt algorithms. An analysis of the algorithms described above shows that for every class $\mathscr{C}$ of locally bounded f we obtain an fpt algorithm for $p\text{-MC}(\text{FO},\mathscr{C})$ with exponent $c + 1$, where c is the exponent of an fpt algorithm for $p\text{-MC}(\text{FO}, f)$. Hence for classes of locally bounded branch width, this yields a quadratic fpt algorithm, and for classes of locally bounded rank width, it yields an fpt algorithm with exponent four.

For classes $\mathscr{C}$ of locally bounded branch width, the exponent can be brought arbitrarily close to 1; more precisely, for every $\varepsilon > 0$ there is an fpt algorithm for $p\text{-MC}(\text{FO},\mathscr{C})$ with a running time of $f(k) \cdot |G|^{1+\varepsilon}$ [43]. Note that we cannot hope to find an fpt algorithm that is linear in the order for general classes of locally bounded branch width, because by Example 4.7 there are classes $\mathscr{C}$ of locally bounded branch width and unbounded average degree, which implies that the size of the graphs in $\mathscr{C}$ is not linearly bounded in the order (and thus an algorithm that is linear in the order cannot even read the whole input graph). It is an open question whether for every class $\mathscr{C}$ of graphs of locally bounded branch width there is an fpt algorithm $p\text{-MC}(\text{FO},\mathscr{C})$ that is linear in the size $||G||$ of the input graph.

For specific classes $\mathscr{C}$, such as the class of planar graphs and classes of bounded genus or bounded degree, it is known that there are fpt algorithms that are linear in the order [43, 75].

Finally, let us look at the parameter dependence of the fpt algorithms. In general, it is again nonelementary by Theorem 3.2, because our classes contain the class of all trees. However, classes of graphs of bounded degree do not contain all trees, and it turns out that for such classes there are fpt algorithms with an elementary parameter dependence. For the class $\mathscr{D}_k$ of graphs of degree at most $k \geq 3$, there is a linear fpt algorithm for $p\text{-MC}(\text{FO},\mathscr{D}_k)$ with a triply exponential parameter dependence, and there is a matching lower bound, which even holds on labelled binary trees [44]. ⌟

5 Digression: Graph minor theory

A graph H is a *minor* of a graph G if H can be obtained from G by deleting vertices, deleting edges, and contracting edges. *Contracting* an edge means removing the edge, identifying its two end vertices, and possibly removing the resulting parallel edges. Figure 5.1 illustrates this. We write $H \preceq G$ if H is isomorphic to a minor of G. A *minor mapping* from H to G is a mapping μ that associates with each $v \in V(H)$ a connected subgraph $\mu(v) \subseteq G$ and with each $e \in E(H)$ an edge $\mu(e) \in E(G)$ such that:

- for all $v \neq w$, the graphs $\mu(v)$ and $\mu(w)$ are vertex disjoint;

- for all $e = \{v,w\} \in E(H)$, the edge $\mu(e)$ is incident to a vertex $v' \in V(\mu(v))$ and a vertex $w' \in V(\mu(w))$.

It is easy to see that $H \preceq G$ if and only if there is a minor mapping from H to G. Observe that the graphs $\mu(v)$ of a minor mapping μ can be chosen to be trees. If μ is a minor mapping from H to G, we call the graph

$$\mu(H) = \left(\bigcup_{v \in V(H)} V(\mu(v)),\ \bigcup_{v \in V(H)} E(\mu(v)) \cup \{\mu(e) \mid e \in E(H)\} \right)$$
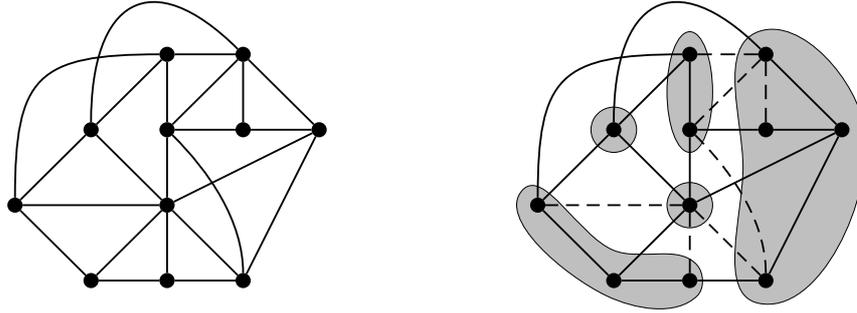
Figure 5.2. An image of K_5 in a nonplanar graph

an *image* of H in G.[6] Figure 5.2 shows an example.

For every graph H, we let

$$\mathscr{X}(H) = \{G \mid H \npreceq G\}.$$

We say that a class $\mathscr{C}$ of graphs *excludes* H if $\mathscr{C} \subseteq \mathscr{X}(H)$. For a class $\mathscr{H}$ of graphs, we let

$$\mathscr{X}(\mathscr{H}) = \bigcap_{H \in \mathscr{H}} \mathscr{X}(H) = \{G \mid H \npreceq G \text{ for all } H \in \mathscr{H}\}.$$

A class $\mathscr{C}$ of graphs is *minor-closed* if for every graph $G \in \mathscr{C}$ and every $H \preceq G$ it holds that $H \in \mathscr{C}$. Observe that a class $\mathscr{C}$ of graphs is minor-closed if and only if it can be *defined by excluding minors*, that is, there is a class $\mathscr{H}$ such that $\mathscr{C} = \mathscr{X}(\mathscr{H})$ (just take $\mathscr{H} = \mathscr{G} \setminus \mathscr{C}$). Robertson and Seymour proved that every minor-closed class of graphs can actually be defined by excluding finitely many minors:

Graph Minor Theorem (Robertson and Seymour [73]). *For every minor-closed class $\mathscr{C}$ of graphs there is a finite class $\mathscr{F}$ of graphs such that*

$$\mathscr{C} = \mathscr{X}(\mathscr{F}).$$

Many natural classes of graphs are minor-closed:

Example 5.1. Every cycle can be contracted to a triangle K_3. Hence the class of forests (acyclic graphs) is precisely $\mathscr{X}(K_3)$. ⌟

Example 5.2. For every $k \geq 1$, the class $\mathscr{B}_k$ of all graphs of branch width k is minor-closed. Let me suggest it as an exercise for the reader to prove this. Furthermore, it holds that $\mathscr{B}_2 = \mathscr{X}(K_4)$ [70]. ⌟

Example 5.3. *Series-parallel graphs* and *outerplanar graphs* exclude K_4. It can be shown that $\mathscr{X}(K_4)$ is precisely the class of all graphs that are subgraphs of series-parallel graphs (see [25], Exercise 7.32). $\mathscr{X}(\{K_4, K_{2,3}\})$ is the class of outerplanar graphs (see [25], Exercise 4.20). ⌟

Example 5.4. By Kuratowski's well-known theorem [54] (or, more precisely, by a variant due to Wagner [81]), the class of planar graphs is $\mathscr{X}(\{K_5, K_{3,3}\})$. ⌟

Example 5.5. For every $k \geq 0$, the class of all graphs of genus k is minor-closed. ⌟

Note that all previous examples of minor-closed classes also have locally bounded branch width. But this is a coincidence, as the following example shows.

Example 5.6. A graph G is an *apex graph* if there is a vertex $v \in V(G)$ such that $G \setminus \{v\}$ is planar. The class of all apex graphs is minor-closed.

---

[6]In the literature, the term "model" is used instead of "image". We prefer "image" here to avoid confusion with "models" in the logical sense.

The class of apex graphs does not have locally bounded branch width. To see this, consider the "pyramid graphs" P_n obtained from the $(n \times n)$-grid $G_{n \times n}$ by adding a new vertex and connecting it to all vertices of the grid. Obviously, the pyramid graphs are apex graphs, and for every $n \geq 1$ we have

$$\ell_{\mathrm{bw}}(P_n, 1) \geq \mathrm{bw}(G_{n \times n}) \geq n,$$

where the second inequality holds by Example 3.10. ⌋

Example 5.7. A graph is *knot free* if it can be embedded into $\mathbb{R}^3$ in such a way that no cycle of the graph is knotted in a nontrivial way. It is easy to see that the class of all knot free graphs is minor-closed.

Similarly, the class of all graphs that can be embedded into $\mathbb{R}^3$ in such a way that no pair of cycles is linked is minor-closed. ⌋

Let me also mention a "non-example": The class of all graphs of crossing number $k \geq 1$ is *not* minor-closed.

5.1 Structure theory

The proof of the graph minor theorem relies on a deep structure theory for classes of graphs with excluded minors. While it is far beyond the scope of this survey to describe this theory in adequate detail, or even give a precise statement of the main structural result, I would like to give the reader a glimpse of the theory, because the model checking algorithms for graphs with excluded minors heavily rely on it. Let me start with a disclaimer: The following intuitive remarks may make a nice story, but they do not always reflect the actual proofs and thus should be taken with some care.

Suppose we have a class $\mathscr{C}$ with excluded minors. Then $\mathscr{C} \subseteq \mathscr{X}(K_k)$ for some k, because every graph is a minor of some complete graph. We fix $\mathscr{C}$ and k for the rest of this section. We want to describe the structure of the graphs in $\mathscr{C}$ by "decomposing" them into "simple" building blocks. We shall define later what exactly we mean by "decomposing" a graph. For now, let us just remark that if a graph has bounded branch width, then we can decompose it into pieces of bounded size. Thus we are mainly interested in classes $\mathscr{C}$ of unbounded branch width. The following theorem, which is one of the fundamental results of the whole theory, gives us a handle on the structure of graphs of unbounded branch width:

Excluded Grid Theorem (Robertson and Seymour [69]). *There is a computable function f such that for every $k \geq 1$ and every graph G, if $\mathrm{bw}(G) \geq f(k)$ then $G_{k \times k} \preceq G$.*

A proof of this theorem can be found in [25].

The Excluded Grid Theorem tells us that if our class $\mathscr{C}$ has unbounded branch width, then the graphs in $\mathscr{C}$ contain large grids as minors. Now we can try to use these large grids as "coordinate systems" and describe the structure of the graphs relative to the grids. So suppose we have a graph $G \in \mathscr{C}$ with a large grid minor, and let $H \subseteq G$ be the image of a large grid. Let us further assume that G is highly connected; if it is not we first decompose it into highly connected parts and then consider each of them separately. We come back to this decomposition process later. We think of the grid as embedded into the plane and the rest of G being glued onto H. It can be proved now that $G \setminus H$ must be glued onto H in a fairly "orderly" way: If there are many pairwise far apart "crossings" in the interior of G then we can find a K_k-minor in G, which is impossible because $G \in \mathscr{C} \subseteq \mathscr{X}(K_k)$. Here a crossing consists of two pairwise disjoint paths with endpoints v_1, v_3 and v_2, v_4 respectively, such that v_1, v_2, v_3, v_4 occur in this clockwise order on some cycle of the grid. Figure 5.3 shows a grid with two crossings. This leaves us with the following structure: There is a bounded number of vertices, called *apices*, that are connected to the grid in an arbitrary fashion. After removing the apices, there still may be many crossings, but they must be grouped together into a bounded number of small regions, called *vortices*. Apart from the apices and the vortices, the rest of G must fit nicely into the planar structure of the grid, that is, the components of $G \setminus H$ are planar pieces, each of which can be embedded into a "square" of the grid. However, so far we have only talked about the interior of the grid. There may be connections between different parts of the exterior cycle of the grid, but they cannot be too wild either, because otherwise we could find a large clique minor again. We can subdivide the exterior cycle into a bounded number of segments and stick some of these together. This gives us a graph that can be embedded into a surface of bounded genus (recall that every surface can be obtained by
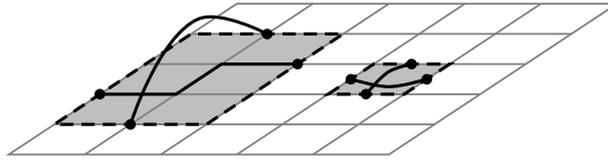
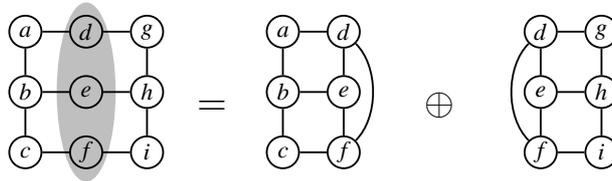**Figure 5.3.** A grid with two crossings

Figure 5.4. A clique sum

gluing together edges of a convex polygon in the plane). Thus after removing a bounded number of apices and vortices, G can be embedded into a surface of bounded genus. We say that G has *almost bounded genus*. We assumed that G is highly connected; if it is not then we can decompose it into pieces with this property. This is Robertson and Seymour's main structure theorem [72]: *For every class $\mathscr{C}$ of graphs with an excluded minor, the graphs in $\mathscr{C}$ can be decomposed into graphs that have almost bounded genus.*

Let us now make it precise what we mean by "decomposing" a graph. Intuitively, we want to recursively split the graph along small separators until there no longer are small separators and the graph is highly connected. But if we do this, we lose too much structure in the decomposition process, because two vertices that are far apart on one side of the partition may be close together on the other side and hence in the original graph. Thus "locality", and similarly "connectivity", may be destroyed in the decomposition process, and this is something we would like to avoid. We take a very drastic approach: Whenever we separate a graph, on both sides we add edges between all vertices in the separator.

We call a graph G a *clique sum* of graphs G_1 and G_2 (and write $G = G_1 \oplus G_2$) if $G_1 \cap G_2$ is a complete graph, $V(G) = V(G_1) \cup V(G_2)$, $E(G) \subseteq E(G_1) \cup E(G_2)$, and $E(G_1) \setminus E(G) \subseteq E(G_2)$, $E(G_2) \setminus E(G) \subseteq E(G_1)$. Thus G is a subgraph of $G_1 \cup G_2$ obtained by possibly deleting some of the edges in $G_1 \cap G_2$. Figure 5.4 illustrates this. Note that we are slightly abusing notation here because there may be several non-isomorphic graphs G such that $G = G_1 \oplus G_2$.

A *clique sum decomposition* of a graph G is a pair (T, γ) consisting of a binary tree T and a mapping γ that associates a graph $\gamma(t)$ with every node $t \in V(T)$ such that $\gamma(r) = G$ for the root r of T and $\gamma(t) = \gamma(t_1) \oplus \gamma(t_2)$ for all nodes t with children t_1, t_2. Figure 5.5 shows an example of a clique sum decomposition of a graph. The decomposition in Figure 5.5 is *complete* in the sense that the graphs at the leaves cannot be decomposed any further. In general, the clique sum decompositions we are interested in are not necessarily complete.

We call the graphs $\gamma(t)$ in a clique sum decomposition (T, γ) the *parts* of the decomposition and the parts $\gamma(t)$ for the leaves t the *atomic parts*, or just *atoms*. (T, γ) is a clique sum decomposition *over* a class $\mathscr{A}$ of graphs if all atoms of (T, γ) belong to $\mathscr{A}$. We call a graph *decomposable over $\mathscr{A}$* if it has a clique sum decomposition over $\mathscr{A}$ and denote the class of all graphs that are decomposable over $\mathscr{A}$ by $\mathscr{D}(\mathscr{A})$.
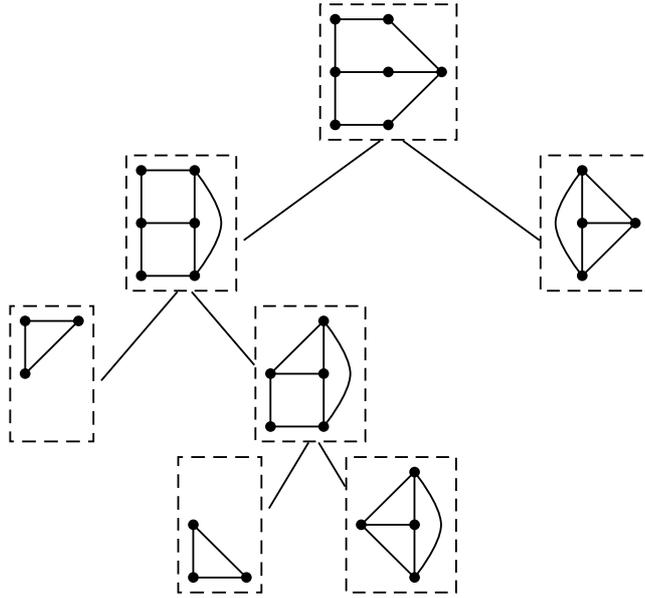
Figure 5.5. A clique sum decomposition

Example 5.8. Let $k \geq 1$, and let $\mathcal{O}_k$ be the class of all graphs of order at most k. If a graph G is decomposable over $\mathcal{O}_k$, then $\mathrm{bw}(G) \leq \max\{k, 2\}$. Let me suggest it as an exercise for the reader to verify this simple fact.

Conversely, it is not too hard to prove that if a graph has branch width at most k, then it is decomposable over $\mathcal{O}_{\lceil (3/2) \cdot k \rceil}$.

Let me remark that a graph has tree width k if and only if it is decomposable over $\mathcal{O}_{k+1}$. This follows from the fact that a graph has tree width at most k if and only if it is a subgraph of a chordal graph of clique number $k+1$ (see Corollary 12.3.12 of [25]). The result for branch width then follows by (3.2). ⌟

I leave it as an exercise to prove the following simple lemma:

Lemma 5.9. *If a class $\mathscr{A}$ of graphs is minor-closed, then the class $\mathscr{D}(\mathscr{A})$ is also minor-closed.*

Robertson and Seymour's structure theorem for classes of graphs with excluded minors can now be stated slightly more precisely as follows: *For every class $\mathscr{C}$ of graphs with an excluded minor there is a class $\mathscr{A}$ of graphs that have almost bounded genus such that $\mathscr{C} \subseteq \mathscr{D}(\mathscr{A})$.* Of course this still leaves it open what exactly is meant by "almost bounded genus". We refer the curious reader to the last chapter of Diestel's book [25] for a more comprehensive introduction to the theory, or to Robertson and Seymour's original article [72].

We close this section by stating a simplified version of a Robertson and Seymour's structure theorem that will be sufficient for our purposes. Recall that ℓ_{bw} denotes the localization of branch width. Minor-closed classes of locally bounded branch width are particularly well behaved. Eppstein [34, 35] proved that a minor closed class $\mathscr{C}$ has locally bounded branch width if and only if it does not contain all apex graphs (recall the definition of apex graphs from Example 5.6). Demaine and Hajiaghayi [22] proved that if a class of graphs has locally bounded branch width, then there actually is a linear bound on the local branch width, that is, there is a $\lambda \geq 1$ such that for all $G \in \mathscr{C}$ and for all $r \geq 1$ it holds that $\ell_{\mathrm{bw}}(G, r) \leq \lambda \cdot r$. This motivates the definition of the following classes of graphs, for every $\lambda \geq 1$:

$$\mathscr{L}_\lambda = \left\{ G \mid \ell_{\mathrm{bw}}(H, r) \leq \lambda \cdot r \text{ for all } H \preceq G \right\}.$$

For every $\mu \geq 0$, we define a class of graphs that are "μ-close" to $\mathscr{L}_\lambda$:

$$\mathscr{L}_{\lambda,\mu} = \left\{ G \mid \exists X \subseteq V(G) : |X| \leq \mu \text{ and } G \setminus X \in \mathscr{L}_\lambda \right\}.$$

Theorem 5.10 (Grohe [48]). *For every class $\mathscr{C}$ with excluded minors, there exist nonnegative integers λ, μ such that*

$$\mathscr{C} \subseteq \mathscr{D}(\mathscr{L}_{\lambda, \mu}).$$

To obtain this result from Robertson and Seymour's structure theorem, one only has to prove that graphs of almost bounded genus are in $\mathscr{L}_{\lambda, \mu}$ for suitable λ, μ. This is not very difficult.

5.2 Algorithms

Before we get back to model checking problems, let me briefly describe some other algorithmic applications of graph minor theory. Consider the following two parameterized problems:

p-DISJOINT-PATHS
  Instance: A graph G and vertices $s_1, t_1, \ldots, s_k, t_k \in V(G)$.
Parameter: k.
  Problem: Decide if there are pairwise disjoint paths P_i, for $i \in [k]$, from s_i to t_i in G.

p-MINOR
  Instance: Graph G, H.
Parameter: $|H|$.
  Problem: Decide if $H \preceq G$.

For neither of the two problems, it is even obvious that they belong to the class XP, that is, can be solved in polynomial time for fixed k, $|H|$, respectively. For DISJOINT-PATHS, this was a long standing open problem posed by Garey and Johnson [46]. Robertson and Seymour proved that both problems are fixed-parameter tractable:

Theorem 5.11 (Robertson and Seymour [71]).
p-DISJOINT-PATHS *and* p-MINOR *have cubic fpt algorithms.*

The reader may wonder why we combine both problems in one theorem. The reason is that they are both special cases of the more general *rooted minor problem*. A *rooted graph* is a tuple $(G, v_1, \ldots, v_k)$, where G is a graph and $v_1, \ldots, v_k \in V(G)$, and a rooted graph $(H, w_1, \ldots, w_k)$ is a *rooted minor* of a rooted graph $(G, v_1, \ldots, v_k)$ if there is a minor map μ from H into G such that $v_i \in V(\mu(w_i))$ for all $i \in [k]$. The parameterized problem p-ROOTED-MINOR is defined as p-MINOR, but for rooted graphs. I leave it to the reader to reduce p-DISJOINT-PATHS to p-ROOTED-MINOR. Robertson and Seymour proved that p-ROOTED-MINOR has a cubic fpt algorithm.

To get an idea of the proof it is easiest to look at the disjoint paths problem. Suppose we are given a graph G and $s_1, t_1, \ldots, s_k, t_k \in V(G)$. Let us further assume, to simplify the presentation, that G is $2k$-connected. If $K_{3k} \preceq G$, then we know that there are disjoint paths from the s_is to the t_is: As the graph is $2k$-connected, by Menger's theorem we can find disjoint paths from $s_1, t_1, \ldots, s_k, t_k$ to an image of K_{3k}. Then in the image of K_{3k}, we can connect the pieces in the right way because all connections are there. This is not entirely trivial, because we only have an image of K_{3k} and not a subgraph, but it can be done. So now we can assume that $K_{3k} \not\preceq G$, and we can apply the structure theory for graphs with excluded K_{3k}. If the branch width of G is bounded, we can solve the disjoint paths problem easily, for example, by applying Courcelle's theorem. If the branch width is large, then by the Excluded Grid Theorem, we can find a large grid in G. By the arguments described above, we can now find a small set of vertices such that after removing these vertices, the whole graph G fits nicely into the planar structure of the grid. Passing to a smaller grid if necessary, we may assume that all the s_i and t_i are outside the grid. Now it can be proved that if there are disjoint paths from s_i to t_i for all $i \in [k]$, then there are such paths that avoid the middle vertex of the grid (say, the grid has odd order). Intuitively, it is plausible that if we have a very large grid and k disjoint paths traversing the grid, then we can always re-route them to avoid the middle vertex. Proving this formally turns out to be the most difficult part of the whole proof [66, 67]. It builds on the full structure theory described in the previous section. However, once this is done, we know that we can delete

the middle vertex of the grid and obtain a smaller graph G' such that there are disjoint paths from s_i to t_i for all $i \in [k]$ in G if and only if there are such paths in G'. We repeatedly delete "irrelevant" vertices this way until we obtain a graph of bounded branch width, and then we solve the problem on this graph. This completes our outline of the proof of Theorem 5.11.

Combined with the Graph Minor Theorem, Theorem 5.11 has the following stunning consequence.

Corollary 5.12. *Every minor-closed class $\mathscr{C}$ of graphs is decidable in cubic time.*

Note that a priori there is no reason why every minor-closed class $\mathscr{C}$ of graphs should be decidable at all.

Remarkably, Corollary 5.12 just claims the existence of algorithms, without actually giving us the algorithms. For example, by Example 5.7 it implies the existence of a cubic time algorithm for deciding whether a graph is knot free. But we still do not know such an algorithm! The reason is that we do not know a finite family of excluded minors defining the class of knot free graphs. Corollary 5.12 is constructive in the sense that if we are given a finite family of excluded minors that defines the class $\mathscr{C}$, then we can construct a cubic time algorithm deciding $\mathscr{C}$. However, for many minor-closed classes we do not know such a finite family.

In recent years, there has been a substantial body of work on algorithms for graph problems restricted to graph classes with excluded minors or even generalisations of such classes [1, 21, 23, 24, 48, 53]. The algorithmic meta theorems presented in the following section should be seen in this context as an attempt to get a more global view on the potentials of algorithmic graph minor theory.

We close this section with a lemma that we will need in the next section.

Lemma 5.13. *For every minor-closed class $\mathscr{A}$ of graphs there is an algorithm that, given a graph $G \in \mathscr{D}(\mathscr{A})$, computes a clique sum decomposition of G over $\mathscr{A}$ in time $O(n^5)$.*

Note that, in particular, the lemma implies an algorithmic version of Theorem 5.10: For every class $\mathscr{C}$ with excluded minors there is a polynomial time algorithm that, given a graph in $\mathscr{C}$, computes a clique sum decomposition of G over $\mathscr{L}_{\lambda,\mu}$.

Proof sketch of Lemma 5.13. Recall that if we write $G = G_1 \oplus G_2$, this implies that $V(G_1 \cap G_2)$ induces a clique in both G_1 and G_2, but not necessarily in G. If it also induces a clique in G, and hence $G = G_1 \cup G_2$, we call the clique sum *simplicial*. We call a clique sum decomposition (T, γ) a *simplicial decomposition* if the clique sums at all nodes of T are simplicial. We call a simplicial decomposition *complete* if its atoms can not be decomposed any further. Simplicial decompositions are much easier to handle than clique sum decompositions. Tarjan [77] showed that a separating clique of a graph can be found in quadratic time. This implies that a complete simplicial decomposition of a graph can be found in cubic time.

Observe that if a graph G has a clique sum decomposition over $\mathscr{A}$, then some supergraph $G' \supseteq G$ with the same vertex set has a simplicial decomposition over $\mathscr{A}$. As $\mathscr{A}$ is closed under taking subgraphs, we may actually assume that this simplicial decomposition is complete.

To compute a clique sum decomposition of a graph G over $\mathscr{A}$, we proceed as follows: We add a maximal set of edges to G so that the resulting graph G' is still in the class $\mathscr{D}(\mathscr{A})$. We can do this in time $O(n^5)$, testing membership in the minor-closed class $\mathscr{D}(\mathscr{A})$ in cubic time for every potential edge. Then we compute a complete simplicial decomposition of the graph G'. This also gives us a clique sum decomposition of G. $\qquad\square$

6 First-order logic on graph classes with excluded minors

Let $\mathscr{C}$ be a class of graphs with excluded minors. Our goal is to design an fpt algorithm for the first-order model checking problem on $\mathscr{C}$. Recall that by Theorem 5.10, the graphs in $\mathscr{C}$ are decomposable into graphs that "almost" have locally bounded branch width, where almost means after removing a bounded number of vertices. We know how to deal with graphs of locally bounded branch width, and it is not hard to extend this to graphs of almost locally bounded branch width. Moreover, we know how to deal with tree structured graphs. By combining these things, so it seems, it should not be too hard to obtain the desired result. This is true, but there are technical difficulties to overcome.

We say that a tuple $\bar{v}$ of vertices of a graph G *induces a clique* in G if $G[\{\bar{v}\}]$ is a complete graph. We write $G = G' \oplus_{\bar{v}} H$ to denote that G is a clique sum of graphs G' and H with $V(G') \cap V(H) = \{\bar{v}\}$. For tuples $\bar{v}_1, \ldots, \bar{v}_m$ of vertices in G' and graphs $H_1, \ldots, H_m$, we may write $G' \oplus_{\bar{v}_1} H_1 \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m$; the order of the summation of the H_is does not matter. In the following, types are always first-order types, and we write tp instead of tp^{FO}. Let me remark that of the two lemmas below that are concerned with computing types, Lemma 6.1 also holds for MSO-types instead of FO-types, whereas the Lemma 6.2 only holds for FO-types.

To see that the parameterized problems in Lemmas 6.1 and 6.2 are well-defined, suppose that we have labelled graphs $G, G', H_1, \ldots, H_m$ and tuples $\bar{v}_0, \ldots, \bar{v}_m$ of vertices of G' such that $G = G' \oplus_{\bar{v}_1} H_1 \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m$. Then it follows from Lemma 2.3 that $\mathrm{tp}_q(G, \bar{v}_0)$ only depends on the types $\mathrm{tp}_q(H_1, \bar{v}_1), \ldots, \mathrm{tp}_q(H_m, \bar{v}_m)$ and not on the actual graphs H_i. That is, for all graphs $H_1', \ldots, H_m'$ with $V(G' \cap H_i') = \{\bar{v}_i\}$ and $\mathrm{tp}_q(H_i', \bar{v}_i) = \mathrm{tp}_q(H_i, \bar{v}_i)$ it holds that

$$\mathrm{tp}_q(G' \oplus_{\bar{v}_1} H_1' \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m') = \mathrm{tp}_q(G, \bar{v}_0).$$

Lemma 6.1. *The following problem is fixed parameter tractable:*

> *Instance:* A labelled graph G' of branch width k, tuples $\bar{v}_i \in V(G')^{k_i}$ for $i \in [0, m]$ that induce cliques in G', and q-types $\Theta_1, \ldots, \Theta_m$.
> *Parameter:* q.
> *Problem:* Compute the type $\mathrm{tp}_q(G, \bar{v}_0)$ for all graphs $G = G' \oplus_{\bar{v}_1} H_1 \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m$, where the H_i are graphs with $\mathrm{tp}_q(H_i, \bar{v}_i) = \Theta_i$ for all $i \in [m]$.

Proof sketch. The proof is similar to the second proof of Courcelle's Theorem: We take a branch decomposition of G'. Starting at the leaves, we compute the types of the boundaries of all nodes. To accomodate for the graphs H_i, we label some of the leaves of the branch decomposition with the cliques $\bar{v}_i$, for $i \in [m]$, instead of edges of G'. The type that is passed from such a leaf to its parent in the computation is Θ_i. In order to obtain the type $\mathrm{tp}_q(G, \bar{v}_0)$ and not just $\mathrm{tp}_q(G, ())$ (the type of the empty tuple) at the root, at each node t of the decomposition we compute the type of a tuple consisting of the vertices in the boundary $\partial\tilde{\beta}(t)$ together with all vertices of the subgraph $G'[\![\tilde{\beta}(t)]\!]$ that appear in the tuple $\bar{v}_0$ (instead of just the vertices in $\partial\tilde{\beta}(t)$). $\square$

Lemma 6.2. *For all λ, μ, the following problem is fixed-parameter tractable:*

> *Instance:* A labelled graph $G' \in \mathcal{L}_{\lambda,\mu}$, tuples $\bar{v}_i \in V(G')^{k_i}$ for $i \in [0, m]$ that induce cliques in G', and q-types $\Theta_1, \ldots, \Theta_m$.
> *Parameter:* q.
> *Problem:* Compute the type $\mathrm{tp}_q(G, \bar{v}_0)$ for all graphs $G = G' \oplus_{\bar{v}_1} H_1 \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m$, where the H_i are graphs with $\mathrm{tp}_q(H_i, \bar{v}_i) = \Theta_i$ for all $i \in [m]$.

Proof sketch. We prove the statement by induction on μ. For $\mu = 0$, that is, graphs in $\mathcal{L}_\lambda$, it can be proved similarly to Theorem 4.8 (using Lemma 6.1 locally).

So let $\mu > 0$. Suppose we are given an instance of the problem. We observe that the graph G' contains a vertex w such that $G' \setminus \{w\} \in \mathcal{L}_{\lambda,\mu-1}$. As $\mathcal{L}_{\lambda,\mu-1}$ is minor-closed and hence decidable in cubic time by Corollary 5.12, we can find such a vertex in time $O(n^4)$. We define a new labelled graph G^* by deleting the vertex w and labelling all vertices adjacent to w in G' with a new label P. We then translate every formula ψ of quantifier rank at most q into a formula ψ^* such that $G \models \psi(\bar{v}_0) \iff G^* \models \psi^*(\bar{v}_0)$. As $G^* \in \mathcal{L}_{\lambda,\mu-1}$, we can apply the induction hypothesis to check if $G^* \models \psi^*(\bar{v}_0)$, and this way we can compute the type of $\bar{v}_0$ in G. $\square$

Theorem 6.3 (Flum and Grohe [39]). *For every class $\mathscr{C}$ of graphs with an excluded minor, the problem $p\text{-MC}(\mathrm{FO}, \mathscr{C})$ is fixed-parameter tractable.*
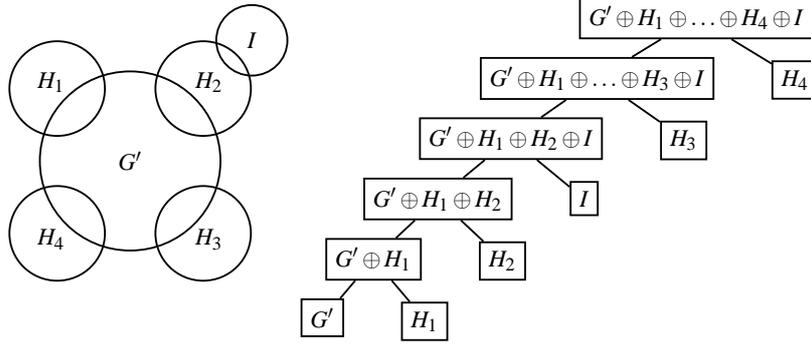
Figure 6.1. The left hand side shows a graph and the right hand side a clique sum decomposition of this graph where the atom G' intersects four other atoms and the atom H_2 intersects two other atoms

Proof sketch. Let $G \in \mathscr{C}$ and $\varphi \in$ FO, say, of quantifier rank q.

Let $\lambda, \mu \geq 0$ such that $\mathscr{C} \subseteq \mathscr{D}(\mathscr{L}_{\lambda,\mu})$. Using Lemma 5.13, we compute a clique sum decomposition (T, γ) of G over $\mathscr{L}_{\lambda,\mu}$.

Now the obvious idea is to compute the q-types of the "boundary tuples" for the parts $\gamma(t)$ in the decomposition in a bottom-up fashion, similarly to the second proof of Courcelle's Theorem. Unfortunately, this simple idea does not work, because a clique sum decomposition is not as well-behaved as a branch decomposition, and the boundaries of the parts may have unbounded size. It may even happen that an atom of the decomposition (corresponding to a leaf of the tree) intersects all other atoms. Figure 6.1 illustrates this.

Observe that a graph in $\mathscr{L}_{\lambda,\mu}$ cannot contain a clique with more than $k = \lceil (3/2) \cdot \lambda + \mu \rceil$ vertices. Hence for all nodes t of T with children t_1, t_2, we must have $V(\gamma(t_1) \cap \gamma(t_2)) \leq k$, because $V(\gamma(t_1) \cap \gamma(t_2))$ is a clique in the $\gamma(t_i)$, and this clique will appear in some atom of the decomposition. Let us fix some order of the vertices of G. For every inner node t with children t_1, t_2, we let $\bar{c}_t$ be the ordered tuple that contains the elements of $V(\gamma(t_1) \cap \gamma(t_2))$.

Our algorithm proceeds recursively, that is, "top-down", instead of "bottom up" as the algorithm in the proof of Courcelle's Theorem, to compute the types of the tuples $\bar{c}_t$. Let us start at the root r of T. Our goal is to compute the q-type of the empty tuple in G. Suppose that the clique sum at r is $G = G_1 \oplus G_2$. We now want to compute the q-type of the tuple $\bar{c}_r$ in both G_1 and G_2; from that we easily get the q-type of the empty tuple in G using Lemma 2.3. So let us continue by computing the q-type of $\bar{c}_r$ in G_1. Suppose the children of t_1 are t_{11} and t_{12}. Let $\bar{c}_1 = \bar{c}_{t_1}$. Now we have a problem: To determine the q-type of $\bar{c}_r$ in G_1, it does not suffice to compute the q-types of $\bar{c}_1$ in G_{11} and G_{12}, because $\bar{c}_r$ and $\bar{c}_1$ may be disjoint tuples. It seems that we have to compute the q-type of the longer tuple $\bar{c}_1 \bar{c}_r$ in both graphs. But clearly we cannot afford the tuples to get longer at every recursion level. Now recall that $\{\bar{c}_r\}$ is a clique in G_1. Hence it is either contained in $\{\bar{c}_1\} = V(G_{11}) \cap V(G_{12})$, in which case we have no problem anyway, or it is contained in precisely one of the two graphs G_{11}, G_{12}. Suppose $\bar{c}_r$ is contained in G_{12}. Then we first compute the q-type Θ of the tuple $\bar{c}_1$ in G_{11}. Now we have to compute the type of $\bar{c}_r$ in the graph $G_1 = G_{11} \oplus G_{12}$. That is, we are in the situation where we have to compute the type of a tuple $\bar{v}$ of vertices of a graph G' in a graph $G' \oplus_{\bar{v}'} H$ for some (and hence all) graph(s) H with $\mathrm{tp}_q(H, \bar{v}') = \Theta$. Furthermore, we know that $\bar{v}, \bar{v}'$ induce cliques in G'. The general problem we have to solve recursively at all nodes of the decomposition tree is the following:

> Compute the q-type of a tuple $\bar{v}_0$ of vertices of a graph G' in a graph $G' \oplus_{\bar{v}_1} H_1 \oplus_{\bar{v}_2} \ldots \oplus_{\bar{v}_m} H_m$ for some (and hence all) graph(s) H_i with $\mathrm{tp}_q(H_i, \bar{c}_i) = \Theta_i$. Here all the tuples $\bar{v}_i$ have length at most k, and they induce cliques in G'.

At the leaves we can use Lemma 6.2 to do this. At the inner nodes, we proceed as described for the node t_1 above. $\qquad \square$

The proof of the theorem actually shows that for all classes $\mathscr{C}$ with excluded minors, p-MC(FO, $\mathscr{C}$) has an fpt algorithm with exponent at most 5. Hence, the exponent is independent of the class $\mathscr{C}$. Thus we

have "almost" proved that there is an fpt algorithm for the model checking problem parameterized both by formula size and the size of the excluded minor. With considerable additional effort, we can get rid of the "almost" in this statement. Let me explain where the difficulties are and, in very general terms, how they are resolved.

Let us first make the statement precise. We define a new graph invariant *excluded minor order (emo)* by letting

$$\mathrm{emo}(G) = \min\{|H| \mid H \npreceq G\}$$

for every graph G. Note that $\mathrm{emo}(G) = \min\{n \mid K_n \npreceq G\}$ and that a class $\mathscr{C}$ excludes a minor if and only if it has bounded excluded minor order. Our goal is to prove that the following problem is fixed-parameter tractable:

> p-$\mathrm{MC}(\mathrm{FO},\mathrm{emo})$
> *Instance:* A graph G and a sentence $\varphi \in \mathrm{FO}$.
> *Parameter:* $|\varphi| + \mathrm{emo}(G)$.
> *Problem:* Decide if $G \models \varphi$.

We have already proved that for every k there is an fpt algorithm A_k with exponent 5 for the first-order model checking problem on the class of all graphs of excluded minor order at most k. The problem is that the family A_k of algorithms is *nonuniform*, that is, we have a different algorithm for every k. To prove that p-$\mathrm{MC}(\mathrm{FO},\mathrm{emo})$ is fixed-parameter tractable, we need a uniform family A_k, or equivalently, a single algorithm A that takes k as an additional input. The family of algorithms we construct in the proof is nonuniform because we use Corollary 5.12 to get decision algorithms for the minor-closed classes $\mathscr{L}_{\lambda,\mu}$ (in the proof of Lemma 6.2) and $\mathscr{D}(\mathscr{L}_{\lambda,\mu})$ (in the proof of Lemma 5.13) for parameters λ,μ that depend on the excluded minor order of the input graph. If we could compute finite families of excluded minors characterising the classes $\mathscr{L}_{\lambda,\mu}$ and $\mathscr{D}(\mathscr{L}_{\lambda,\mu})$ from the parameters λ,μ, then we would be fine, but we currently do not know how to do this. Fortunately, there is an alternative approach that avoids Corollary 5.12 entirely. The application of Corollary 5.12 in the proof of Lemma 5.13 yielded an algorithm for computing a clique sum decomposition of a graph over $\mathscr{D}(\mathscr{L}_{\lambda,\mu})$. While we do not know how to compute such a decomposition uniformly in λ and μ, in [18] we found a way to compute, uniformly in λ,μ, a decomposition that is a sufficiently good approximation of the desired clique sum decomposition. The algorithm recursively splits the input graph along small separators that are sufficiently "balanced". The application of Corollary 5.12 in the proof of Lemma 6.2 was needed to find a set of at most μ vertices in a graph in $\mathscr{L}_{\lambda,\mu}$ whose removal left a graph in $\mathscr{L}_\lambda$. In [18], we found an fpt algorithm that, given a graph $G \in \mathscr{L}_{\lambda,\mu}$, computes a set $W \subseteq V(G)$ of at most μ vertices such that $G \setminus W \in \mathscr{L}_{\lambda'}$ for some λ' that is effectively bounded in terms of λ. This is good enough for our purposes. Putting everything together, we obtain the following result:

Theorem 6.4 (Dawar, Grohe, and Kreutzer [18]). *p-$\mathrm{MC}(\mathrm{FO},\mathrm{emo})$ is fixed-parameter tractable.*

We say that a class *locally excludes a minor* if it has locally bounded excluded minor order. Then combining Theorems 6.4 and 4.8, we get:

Corollary 6.5 ([18]). *For every class $\mathscr{C}$ locally excluding a minor, the problem p-$\mathrm{MC}(\mathrm{FO},\mathscr{C})$ is fixed-parameter tractable.*

7 Other logics and other problems

In this section, we briefly discuss some extensions of the main results mentioned in this survey to more powerful logics, and also to variants of the basic model checking problem.

7.1 Other logics

It is really not much that is known about algorithmic meta theorems for logics other than first-order and monadic second-order logic. Courcelle's Theorem and its variant for graphs of bounded rank width can be

extended to the extension of monadic second order logic by modulo counting quantifiers [10, 12] (also see [57]), and clearly not to full binary second order logic.

As for the results for first-order logic, let us consider potential extensions of the model-checking results to *monadic transitive closure logic* and *monadic least fixed-point logic*. Both transitive closure logic and least fixed-point logic have been extensively studied in finite model theory [31, 55]. Their monadic fragments are strictly contained in monadic second-order logic, and they strictly contain first-order logic. (When we say that a logic *contains* another logic, we mean semantic containment, that is, L_1 *contains* L_2 if every formula of L_2 is logically equivalent to a formula of L_1. We say that L_1 *strictly contains* L_2 it L_1 contains L_2, but L_2 does not contain L_1.) Monadic transitive closure logic and monadic least fixed-point logic seem to mark the boundary of the range of logics to which the tractability results for first-order model checking can be extended.

Monadic transitive closure logic TC^1 is the extension of first-order logic by formulas of the form $[TC_{x,y}\varphi](x,y)$, where φ is a formula with free variables among $\{x,y\}$. The free variables of the formula $[TC_{x,y}\varphi](x,y)$ are x and y. It is allowed to nest TC-operators arbitrarily and interleave them with first-order quantifiers and connectives. However, we do not allow any other free variables than x and y in the formula φ in $[TC_{x,y}\varphi](x,y)$. The semantics is defined as follows: If G is a (labelled) graph and $v, w \in V(G)$, then $G \models [TC_{x,y}\varphi](v,w)$ if and only if there is an $m \geq 1$ and vertices $v_1, \ldots, v_m \in V(G)$ such that $v = v_1, w = v_m$, and $G \models \varphi(v_i, v_{i+1})$ for all $i \in [m-1]$.

Example 7.1. The following TC^1-sentence states that a graph is connected:

$$\forall x \forall y [TC_{x,y} E(x,y)](x,y).$$

It is known that there is no sentence of first-order logic defining connectivity (see, e.g., [31, 32, 55]). ⌟

Example 7.2. The following TC^1-sentence states that a graph has no cyclic walk of odd length and hence is bipartite

$$\neg \exists x \exists y \Big(\big[TC_{x,y} \exists z \big(E(x,z) \wedge E(z,y) \big) \big] (x,y) \wedge E(y,x) \Big).$$

Again, it is known that there is no sentence of first-order logic defining bipartiteness. ⌟

The logic TC^1 trivially contains FO, and it is strictly contained in MSO. As opposed to MSO, its data complexity is still in polynomial time (actually, in nondeterministic logarithmic space).

Theorem 7.3. *Let $\mathscr{C}$ be a class of graphs that contains all planar graphs of degree at most* 3. *Then p-MC$(TC^1, \mathscr{C})$ is hard for the parameterized complexity class* AW[∗].

Proof sketch. We reduce the model checking problem for first-order logic on arbitrary graphs, which is known to be AW[∗]-complete (by Theorem 2.12), to p-MC$(TC^1, \mathscr{C})$. Let G be a graph and φ a first-order sentence.

We start with constructing a drawing of G in the plane, which of course may involve edge crossings. We can find a drawing with at most polynomially many (in the number of vertices of G) crossings such that in each point of the plane at most 2 edges cross. We introduce five new labels P_1, P_2, Q_1, Q_2, R. We define a new labelled graph G_1 by labelling each vertex of the original graph G with P_1 and replacing each edge crossing in the drawing of G by a little gadget, as shown in Figure 7.1. Observe that the edge relation of the graph G can be defined in G_1 by a TC^1-formula (but not by an FO-formula, because an edge may cross many other edges). G_1 is planar, but may have degree greater than 3. We define a graph G_2 by replacing every vertex v of G_1 of degree d by a binary tree with exactly d leaves. With each leaf we associate one vertex w adjacent to v in G_1. We connect the leaf of the v-tree associated with w with the leaf of the w-tree associated with v. Then we identify v with the root of its tree, label it P_1, and label all other vertices of the tree P_2. Then the edge relation of G is also definable in G_2 by a TC^1-formula. We can use this formula to translate the formula φ into a TC^1-formula φ_2 such that

$$G \models \varphi \iff G_2 \models \varphi_2.$$

G_2 is a planar graph of degree at most 3, and it clearly can be computed from G in polynomial time. This gives us the desired reduction. □
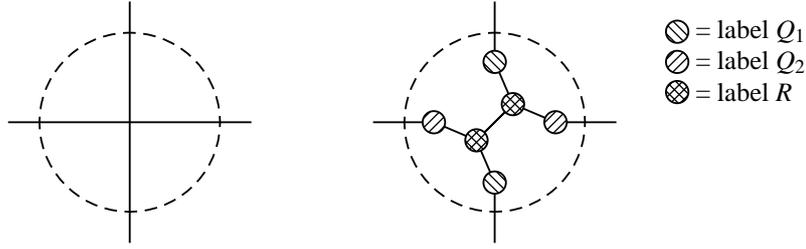
Figure 7.1. A gadget for edge crossings

Monadic least-fixed-point logic LFP^1 (see, e.g., [50, 74]) is the extension of first-order logic by formulas of the form $[\text{LFP}_{x,X}\,\varphi](x)$, where φ is a first-order formula such that X only occurs positively in φ and φ has no free individual variables other than x. (It may have free set variables other than X.) The free variables of $[\text{LFP}_{x,X}\,\varphi](x)$ are x and all free set variables of φ except X. To define the semantics, let $\varphi = \varphi(x, X, Y_1, \ldots, Y_m)$. Let G be a labelled graph and $W_1, \ldots, W_m \subseteq V(G)$, $v \in V(G)$. Then $G \models [\text{LFP}_{x,X}\,\varphi(x, X, W_1, \ldots, W_m)](v)$ if and only if v is in the least fixed point of the monotone operator $U \mapsto \{u \mid G \models \varphi(u, U, W_1, \ldots, W_m)\}$ on $V(G)$. We call a formula in LFP^1 *restricted* if for every subformula of the form $[\text{LFP}_{x,X}\,\varphi](x)$, the formula φ has no free set variables other than X. By LFP^1_r we denote the fragment of LFP^1 consisting of all restricted formulas.

The reason for requiring that a formula φ in the scope of a fixed-point operator $[\text{LFP}_{x,X}\,\varphi](x)$ contains no free individual variables other than x is that otherwise even the restricted fragment of the logic would contain TC^1. It can be shown that LFP^1 (as defined here) does not contain TC^1 and that, conversely, TC^1 does not contain LFP^1, not even LFP^1_r.

I was unable to come up with convincing examples of properties of plain graphs that are definable in LFP^1_r or LFP^1, but not in first-order logic. However, this changes when we admit more general structures. For example, on *Kripke structures*, that is, labelled directed graphs with one distinguished element, LFP^1 contains the modal μ-calculus. Here is another example:

Example 7.4. We can describe monotone Boolean circuits as labelled directed acyclic graphs, and assignments to the input gates by an additional label. It is easy to see that there is an LFP^1_r-formula stating that an assignment satisfies a circuit. This is not definable in first-order logic. ⌟

As we mentioned earlier, almost all results presented in this survey extend to arbitrary structures. In this context, the following tractability result is more interesting than it may seem in a purely graph theoretical context.

Theorem 7.5. *Let $\mathscr{C}$ be a class of graphs such that p-$\text{MC}(\text{FO}, \mathscr{C}_{lb})$ is fixed-parameter tractable. Then p-$\text{MC}(\text{LFP}^1_r, \mathscr{C}_{lb})$ is fixed-parameter tractable.*

Proof sketch. To evaluate a formula of the form $[\text{LFP}_{x,X}\,\varphi](x)$, where $\varphi = \varphi(x, X)$ is first-order, in a graph G, we proceed as follows: We introduce a new label P. Initially, we set $P(G) = \emptyset$. Then we repeatedly compute the set of all $v \in V(G)$ such that $G \models \varphi(v, P(G))$ using an fpt algorithm for p-$\text{MC}(\text{FO}, \mathscr{C}_{lb})$ and set $P(G)$ to be the set of all these vertices. After at most $n = |G|$ steps, the computation reaches a fixed point, which consists precisely of all v such that $G \models [\text{LFP}_{x,X}\,\varphi](v)$. Using this algorithm as a subroutine, we can easily model-check arbitrary sentences in LFP^1_r. ∎

Lindell [56] proved that for the classes $\mathscr{D}_k$ of graphs of degree at most k, the problem p-$\text{MC}(\text{LFP}^1_r, \mathscr{D}_k)$ even has a linear time fpt algorithm.

7.2 Generalised model checking problems

For a formula $\varphi(x_1, \ldots, x_k)$ and a graph G, by $\varphi(G)$ we denote the set of all tuples $(v_1, \ldots, v_k) \in V(G)^k$ such that $G \models \varphi(v_1, \ldots, v_k)$. For every logic L and class $\mathscr{C}$ of graphs, we may consider the following

variants of the model checking problem $p\text{-MC}(\text{L},\mathscr{C})$: The input always consists of a graph $G \in \mathscr{C}$ and a formula $\varphi \in \text{L}$, possibly with free variables. The parameter is $|\varphi|$. The *decision problem* simply asks if $\varphi(G)$ is nonempty. For logics closed under existential quantification, this problem is equivalent to the model checking problem $p\text{-MC}(\text{L},\mathscr{C})$. Therefore, we will not consider it here anymore. The *construction problem* asks for a solution $\bar{v} \in \varphi(G)$ if there exists one. The *evaluation (or listing) problem* asks for all solutions, that is, for the whole set $\varphi(G)$. Finally, the *counting (or enumeration) problem* asks for the number $|\varphi(G)|$ of solutions. All these problems have natural applications.

The results on monadic second-order model checking on graphs of bounded branch width and bounded rank width (Theorems 3.3 and 3.17) can be extended to the corresponding construction and counting problems [3, 15, 38, 41]. For the evaluation problem, the situation is a bit more complicated because the size of the answer $\varphi(G)$ may be much larger than the size of the input (n^k for a graph of order n and a formula with k free variables), hence we cannot expect an algorithm that is fixed-parameter tractable. However, it has been proved that there is a linear time fpt algorithm for this problem if the running time is measured in terms of the input size plus the output size [16, 38]. Recently, it has been shown that there even is such an algorithm that does a linear (in terms of the input size) pre-computation and then produces solutions with delay bounded in terms of the parameter [4, 13].

Frick [41, 42] proved that the construction problem and counting problem for many classes of graphs of locally bounded branch width, including planar graphs and graphs of bounded degree, has a linear fpt algorithm. This is a nontrivial extension of the model checking results. Even for a simple first-order definable counting problem like the parameterized independent set counting problem ("Count the number of independent sets of size k in a graph."), say, on a class of graphs of bounded degree, it is not obvious how to solve it by an fpt algorithm. For the evaluation problem, again there are linear time fpt algorithms if the running time is measured in terms of the input size plus the output size [41]. For classes of graphs of bounded degree, Durand and Grandjean [30] proved that there is an fpt algorithm for the first-order evaluation problem that does a linear pre-computation and then produces solutions with delay bounded in terms of the parameter.

Finally, let us take a brief look at optimisation problems, which play a central role in complexity theory, but have not been studied very systematically in the context of meta theorems. Consider a first-order formula $\varphi(X)$ that is positive in a free set variable X. Such a formula naturally describes a minimisation problem: Given a graph G, find a set $S \subseteq V(G)$ of minimum size such that $G \models \varphi(S)$. Many natural minimisation problems on graphs can be described this way. An example is the minimum dominating set problem, which can be described by the formula $dom(X)$ of Example 2.1. Similarly, formulas $\varphi(X)$ that are negative in X naturally describe maximisation problems. An example is the maximum independent set problem, which is described by the formula $ind(X) = \forall x \forall y (\neg X(x) \vee \neg X(y) \vee \neg E(x,y))$. We call such optimisation problems *first-order definable*. It was proved in [19] that the restriction of a first-order definable optimisation problem to a class of graphs with an excluded minor has a polynomial time approximation scheme, that is, can be approximated in polynomial time to any factor $(1 + \varepsilon)$, where $\varepsilon > 0$.

8 Concluding remarks and open questions

Figure 8.1 gives an overview of the classes of graphs we have studied in this survey. Let me conclude by mentioning a few directions for further research that I find particularly promising:

8.1 Further tractable classes

Many of the classes of graphs considered in this survey, including all classes excluding a minor, have bounded average degree. It may be tempting to conjecture that first-order model checking is tractable on all classes of graphs of bounded average degree, but it is easy to see that this is not the case. As Stephan Kreutzer observed, it is not even the case for classes of bounded maximum average degree, where the *maximum average degree* of a graph G is the maximum of the average degrees of all subgraphs of G. To see this, just observe that model-checking on a graph G can be reduced to model-checking on its incidence graph (i.e., the graph obtained from G by subdividing each edge once), and that every incidence graph has maximum average degree at most 4.

Nešetřil and Ossona de Mendez [59] introduced a property of graph classes that may be viewed as a
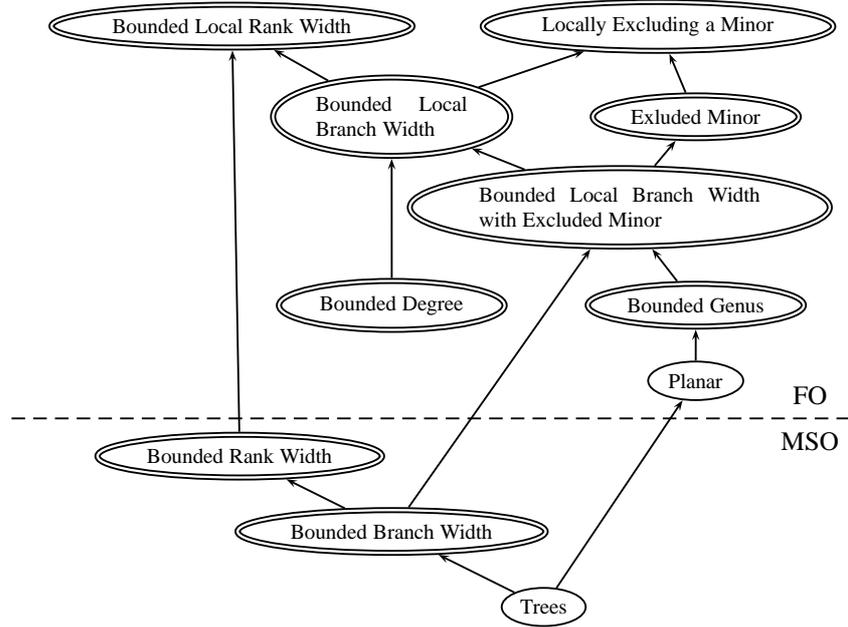
Figure 8.1. Classes of graphs with a tractable first-order model checking problems. Double-lined ellipses contain families of classes. Classes below the dashed line have a tractable monadic second-order model checking problem

refinement of maximum average degree and that avoids such problems. Let G be a graph. The *radius* of a minor mapping μ from a graph H to G is the minimum of the radii of the subgraphs $G[\mu(v)]$, for $v \in V(H)$. We write $H \preceq_r G$ if there is a minor mapping of radius at most r from H to G. Note that $H \preceq_0 G$ if and only if H is a subgraph of G. The *greatest reduced average density (grad) of rank r of G* is the number

$$\nabla_r(G) = \max\left\{ \frac{|E(H)|}{|V(H)|} \,\middle|\, H \preceq_r G \right\}.$$

Note that $\nabla_0(G)$ is half the maximum average degree of G. Now a class $\mathscr{C}$ of graphs has *bounded expansion* if there is some function f such that $\nabla_r(G) \le f(r)$ for all $G \in \mathscr{C}$ and $r \ge 0$. Nešetřil and Ossona de Mendez observed that every class of graphs excluding a minor has bounded expansion. It can be shown that there are classes of bounded expansion that do not exclude a minor, not even locally. Conversely, there are classes of bounded local tree width and hence classes locally excluding a minor that do not have bounded expansion. This follows from Example 4.7 and the fact that classes of bounded expansion have bounded average degree. I refer the reader to [58, 59, 60] for an introduction to classes of bounded expansion and an overview of their nice algorithmic properties.

Open Problem 8.1. *Is $p\text{-MC}(\text{FO},\mathscr{C})$ fixed-parameter tractable for every class $\mathscr{C}$ of graphs of bounded expansion?*

There is no need to restrict the study of structural properties that facilitate efficient model checking to graph theoretic properties such as those predominant in this survey. For example, it would also be very interesting to study the complexity of model-checking problems on finite algebraic structures such as groups, rings, fields, lattices, et cetera.

Open Problem 8.2. *Are $p\text{-MC}(\text{FO},\mathscr{C})$ and $p\text{-MC}(\text{MSO},\mathscr{C})$ fixed-parameter tractable for the classes $\mathscr{C}$ of finite groups, finite abelian groups, finite rings, finite fields?*

8.2 Necessary conditions for tractability

The main results presented in this survey may be viewed as giving sufficient conditions for classes of graphs to have tractable first-order or monadic second-order model checking problems. *What are necessary*

conditions for tractability, and which classes have hard model checking problems? Note that it is not easy to come up with structural conditions for hardness, because we can "cheat" and, for example, pad graphs that have a structure presumably making model checking difficult with a large number of isolated vertices. This makes the model checking problem "easier" just because it gives us more time to solve it. Thus we probably want to impose closure conditions on the classes of graphs we consider, such as being closed under taking subgraphs.

It follows from the Excluded Grid Theorem that for minor-closed classes $\mathscr{C}$ of graphs, $p\text{-MC}(\text{MSO}, \mathscr{C})$ is fixed-parameter tractable if and only if $\mathscr{C}$ has bounded branch width. Actually, this can be slightly strengthened to classes closed under taking topological minors. I do not know of any results beyond that. To stimulate research in this direction, let me state a conjecture:

Conjecture 8.3. *Let $\mathscr{C}$ be a class of graphs that is closed under taking subgraphs. Suppose that the branch width of $\mathscr{C}$ is not poly-logarithmically bounded, that is, there is no constant c such that $\text{bw}(G) \leq \log^c |G|$ for every $G \in \mathscr{C}$.*

Then $p\text{-MC}(\text{MSO}, \mathscr{C})$ is not fixed parameter tractable.

Of course, with current techniques we can only hope to prove this conjecture under some complexity theoretic assumption.

For first-order logic, I have much less intuition. Clearly, the present results are very far from optimal. Just as an illustration, observe that if a class $\mathscr{C}$ of graphs has a tractable first-order model checking problem, then so has the closure of $\mathscr{C}$ under complementation. (Recall that the *complement* of a graph $G = (V, E)$ is the graph $\bar{G} = \left(V, \binom{V}{2} \setminus E\right)$.) However, most of the classes we considered here are not closed under complementation.

8.3 Average Case Analysis

Instead of the worst case running time, it is also interesting to consider the average case. Here even the most basic questions are wide open. For $n \geq 1$, let $\mathscr{W}_n$ be a probability space of graphs with vertex set $[n]$. We say that a model checking algorithm is *fpt on average over* $(\mathscr{W}_n)_{n \geq 1}$ if its expected running time on input $G \in \mathscr{W}_n$ and φ is bounded by $f(|\varphi|) \cdot n^{O(1)}$, for some computable function f. For every function $p : \mathbb{N} \to [0, 1]$ (here $[0, 1]$ denotes an interval of real numbers), let $\mathscr{G}(n, p)$ denote the probability space of all graphs over $[n]$ with edge probability $p(n)$ (see, e.g., [2]). For a constant $c \in [0, 1]$, we let $\mathscr{G}(n, c) = \mathscr{G}(n, p)$ for the constant function $p(n) = c$. In [47], I observed that for $p(n) = \min\{1, c/n\}$, where $c \in \mathbb{R}_{\geq 0}$ is a constant, there is a model checking algorithm for first-order logic that is fpt on average over $(\mathscr{G}(n, p))_{n \geq 1}$.

Open Problem 8.4. *Is there a model checking algorithm for first-order logic that is fpt on average over $(\mathscr{G}(n, 1/2))_{n \geq 1}$?*

Let me suggest it as an exercise for the reader to design a model checking algorithm for existential first-order logic that is fpt on average over $(\mathscr{G}(n, 1/2))_{n \geq 1}$.

8.4 Structures of bounded rank width

Most of the results of this survey can easily be extended from classes $\mathscr{C}$ of graphs to the classes $\mathscr{C}_{str}$ of arbitrary relational structures whose underlying graphs (Gaifman graphs) are in $\mathscr{C}$. However, this is not true for the results that involve rank width. It is not at all obvious what an appropriate notion of rank width for arbitrary structures could look like, and I think it is a challenging open problem to find such a notion.

8.5 Model checking for monadic least fixed-point logic

Conjecture 8.5. *Let $\mathscr{C}$ be a class of graphs such that $p\text{-MC}(\text{FO}, \mathscr{C}_{lb})$ is fixed-parameter tractable. Then $p\text{-MC}(\text{LFP}^1, \mathscr{C}_{lb})$ is fixed-parameter tractable.*

It will be difficult to prove this conjecture, because it is related to the notoriously open problem of whether the model checking problem for the modal μ-calculus is in polynomial time. But maybe the conjecture is wrong; refuting it might be more feasible.

Acknowledgements

References

[1] I. Abraham, C. Gavoille, and D. Malkhi. Compact routing for graphs excluding a fixed minor. In P. Fraigniaud, editor, *Proceedings of the 19th International Conference on Distributed Computing*, volume 3724 of *Lecture Notes in Computer Science*, pages 442–456. Springer-Verlag, 2005.

[2] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 2nd edition, 2000.

[3] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[4] G. Bagan. MSO queries on tree decomposable structures are computable with linear delay. In Z. Ésik, editor, *Proceedings of the 20th International Workshop on Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 167–181. Springer-Verlag, 2006.

[5] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[6] H.L. Bodlaender and D.M. Thilikos. Constructive linear time algorithms for branchwidth. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 627–637. Springer-Verlag, 1997.

[7] Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *Lecture Notes in Computer Science*, pages 109–120. Springer-Verlag, 2006.

[8] B. Courcelle. An axiomatic definition of context-free graph grammars and applications to nlc grammars. *Theoretical Computer Science*, 55:141–181, 1987.

[9] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 194–242. Elsevier Science Publishers, 1990.

[10] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[11] B. Courcelle. The monadic second-order logic of graphs VII: Graphs as relational structures. *Theoretical Computer Science*, 101:3–33, 1992.

[12] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations*, chapter 5, pages 313–400. World Scientific (New-Jersey, London), 1997.

[13] B. Courcelle. Linear delay enumeration and monadic second-order logic, 2006. Available at http://www.labri.fr/perso/courcell/ActSci.html.

[14] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[15] B. Courcelle, J.A. Makowsky, and U. Rotics. On the fixed-parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1–2):23–52, 2001.

[16] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109:49–82, 1993.

[17] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000.

[18] A. Dawar, M. Grohe, and S. Kreutzer. Locally excluding a minor. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer Science*, 2007. To appear.

[19] A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Approximation schemes for first-order definable optimisation problems. In *Proceedings of the 21st IEEE Symposium on Logic in Computer Science*, pages 411–420, 2006.

[20] A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Model theory makes formulas large. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming*, 2007. To appear.

[21] E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, and D.M. Thilikos. Subexponential parameterized algorithms on graphs of bounded-genus and H-minor-free graphs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 830–839, 2004.

[22] E.D. Demaine and M.T. Hajiaghayi. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 840–849, 2004.

[23] E.D. Demaine, M.T. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 637–646, 2005.

[24] E.D. Demaine, M.T. Hajiaghayi, and K.-I. Kawarabayashi. Algorithmic graph minor theory: Improved grid minor bounds and wagner's contraction. In S.K. Madria, K.T. Claypool, R. Kannan, P. Uppuluri, and M.M. Gore, editors, *Proceedings of the Third International Conference on Distributed Computing and Internet Technology*, volume 4317 of *Lecture Notes in Computer Science*, pages 3–15. Springer-Verlag, 2006.

[25] R. Diestel. *Graph Theory*. Springer-Verlag, 3rd edition, 2005.

[26] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24:873–921, 1995.

[27] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141:109–131, 1995.

[28] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[29] R.G. Downey, M.R. Fellows, and U. Taylor. The parameterized complexity of relational database queries and an improved characterization of W[1]. In D.S. Bridges, C. Calude, P. Gibbons, S. Reeves, and I.H. Witten, editors, *Combinatorics, Complexity, and Logic*, volume 39 of *Proceedings of DMTCS*, pages 194–213. Springer-Verlag, 1996.

[30] A. Durand and E.Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Transactions on Computational Logic*. To appear.

[31] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 2nd edition, 1999.

[32] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.

[33] P. Hliněný and S.-I. Oum. Finding branch-decompositions and rank-decompositions. Available at http://www.math.uwaterloo.ca/˜sangil/.

[34] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3:1–27, 1999.

[35] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.

[36] P. Erdös. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959.

[37] R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, pages 43–73, 1974.

[38] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *Journal of the ACM*, 49(6):716–752, 2002.

[39] J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.

[40] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.

[41] M. Frick. *Easy Instances for Model Checking*. PhD thesis, Albert-Ludwigs-Universität Freiburg, 2001.

[42] M. Frick. Generalized model-checking over locally tree-decomposable classes. In H. Alt and A. Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *Lecture Notes in Computer Science*, pages 632–644. Springer-Verlag, 2002.

[43] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48:1184–1206, 2001.

[44] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130:3–31, 2004. LICS 2002 Special Issue.

[45] H. Gaifman. On local and non-local properties. In J. Stern, editor, *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pages 105–135. North Holland, 1982.

[46] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[47] M. Grohe. Generalized model-checking problems for first-order logic. In H. Reichel and A. Ferreira, editors, *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 12–26. Springer-Verlag, 2001.

[48] M. Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003.

[49] M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In C. Beeri and P. Buneman, editors, *Proceedings of the 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 70–82. Springer-Verlag, 1999.

[50] M. Grohe, N. Schweikardt, and S. Kreutzer. The expressive power of two-variable least fixed-point logics. In J. Jedrzejowicz and A. Szepietowski, editors, *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 422–434. Springer-Verlag, 2005.

[51] M. Grohe and S. Wöhrle. An existential locality theorem. *Annals of Pure and Applied Logic*, 129:131–148, 2004.

[52] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

[53] K.-I. Kawarabayashi and B. Mohar. Approximating the list-chromatic number and the chromatic number in minor-closed and odd-minor-closed classes of graphs. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 401–416, 2006.

[54] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[55] L. Libkin. *Elements of Finite Model Theory*. Springer-Verlag, 2004.

[56] S. Lindell. Computing monadic fixed-points in linear-time on doubly-linked data structures, 2005. Available at http://www.haverford.edu/cmsc/slindell/.

[57] J.A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126:159–213, 2004.

[58] J. Nešetřil and P. Ossona de Mendez. Linear time low tree-width partitions and algorithmic consequences. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 391–400, 2006.

[59] J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion I: Decompositions. *European Journal of Combinatorics*, 2007. To appear.

[60] J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion II: Algorithmic aspects. *European Journal of Combinatorics*, 2007. To appear.

[61] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[62] S.-I. Oum. Rank-width is less than or equal to branch-width, 2006. Available at http://www.math.uwaterloo.ca/~sangil/.

[63] S.-I. Oum and P.D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96:514–528, 2006.

[64] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[65] N. Robertson and P.D. Seymour. Graph minors I–XXIII. Appearing in *Journal of Combinatorial Theory, Series B* since 1982.

[66] N. Robertson and P.D. Seymour. Graph minors XXI. Graphs with unique linkages. To appear.

[67] N. Robertson and P.D. Seymour. Graph minors XXII. Irrelevant vertices in linkage problems. To appear.

[68] N. Robertson and P.D. Seymour. Graph minors III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36:49–64, 1984.

[69] N. Robertson and P.D. Seymour. Graph minors V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[70] N. Robertson and P.D. Seymour. Graph minors X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52:153–190, 1991.

[71] N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

[72] N. Robertson and P.D. Seymour. Graph minors XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 77:1–27, 1999.

[73] N. Robertson and P.D. Seymour. Graph minors XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004.

[74] N. Schweikardt. On the expressive power of monadic least fixed point logic. *Theoretical Computer Science*, 350:325–344, 2006.

[75] D. Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6:505–526, 1996.

[76] H. Tamaki. A linear time heuristic for the branch-decomposition of planar graphs. In G. Di Battista and U. Zwick, editors, *Proceedings of the 11th Annual European Symposium on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 765–775. Springer-Verlag, 2003.

[77] R.E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.

[78] J.W. Thatcher and J.B. Wright. Generalised finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2:57–81, 1968.

[79] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[80] M.Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, pages 266–276, 1995.

[81] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114:570–590, 1937.