Chapter 1

INTRODUCTION

In this book we develop a *definable structure theory* for finite graphs. The goal is to decompose graphs into pieces that are "simpler" than the original graphs, and to do this in such a way that the decomposition is definable in some logic. A simple example of a decomposition theorem we prove here is that every graph has a "definable treelike decomposition" into 3-connected graphs. (A graph is 3-connected if it stays connected even after two arbitrary vertices are removed.) A more complicated example states that every graph that excludes K_5, the complete graph on five vertices, as a minor has a definable treelike decomposition into pieces that are either 3-connected planar graphs or isomorphic to the graph L shown in Figure 1.1. (A *minor* of a graph G is a graph H that is obtained from a subgraph of G by contracting edges. We will give more background on graph minors in the next section.)

The main applications of our definable structure theory are in descriptive complexity, and for these applications we need our decompositions to be definable in *least fixed-point logic* LFP, or equivalently, in *inflationary fixed-point logic* IFP. (For technical reasons, it will be more convenient for us to work with IFP.) These fixed-point logics are extensions of first-order predicate logic by fixed-point operators that allow it to formalise inductive definitions.

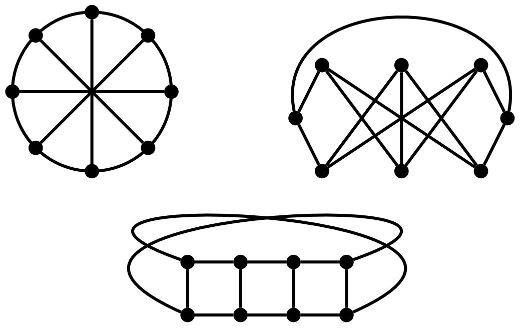

FIGURE 1.1. The graph L drawn in three different ways.

In this book we will exclusively study decompositions definable in IFP, but much of the general theory we develop here applies to other logics as well, for example, to monadic second-order logic.

There is a standard graph-theoretic notion of *tree decomposition*, playing a central role in modern graph structure theory. Ideally, we would like our definable decompositions to be tree decompositions, but it turns out that in general tree decompositions are not logically definable, because they are not invariant under automorphisms of the underlying graph. Instead, we introduce a new notion of *treelike decompositions*. Treelike decompositions inherit many of the desirable properties of tree decompositions, yet they can be made automorphism-invariant and, as it turns out, are often definable in logics like IFP.

Our main theorem, the Definable Structure Theorem 17.2.1, says that all classes of graphs that exclude some fixed graph as a minor admit IFP-definable treelike decompositions into pieces that admit an IFP-definable linear order. Linearly ordered finite graphs are easy to deal with in many ways. For example, they have trivial automorphism groups. More importantly, many results in descriptive complexity require structures to be linearly ordered. It is a long-standing open question whether there is a logical characterisation of the polynomial-time properties of graphs. As an application of our definable structure theorem, we obtain such a characterisation for all properties of graphs with excluded minors. As a second important application of our structure theorem, we show that for every class of graphs that exclude some fixed graph as a minor there is a k such that a simple combinatorial algorithm, namely "the k-dimensional Weisfeiler–Leman algorithm", decides isomorphism of graphs in $\mathcal{C}$ in polynomial time.

The rest of this introductory chapter is structured as follows. In the next section, we describe the graph-theoretic context of our results. After that, we briefly (and informally) explain the central concept of treelike decompositions of graphs. Then we say more about the applications in descriptive complexity theory and to the graph isomorphism problem. Finally, we give an outline of the rest of this book and of the proof of our main theorem, and close the chapter with a few bibliographical remarks.

1.1. Graph minor theory

Recall that a graph H is a minor of a graph G if H is obtained from a subgraph of G by contracting edges. (Formally, contracting an edge means deleting the edge and identifying its endvertices.) Figure 1.2 shows an example. If $\mathcal{C}$ is a class of graphs such that H is not a minor of any $G \in \mathcal{C}$, then we say that $\mathcal{C}$ *excludes H as a minor*. Graph minor theory is concerned with graph classes excluding some fixed graph as a minor. The starting point of the theory is perhaps a variant

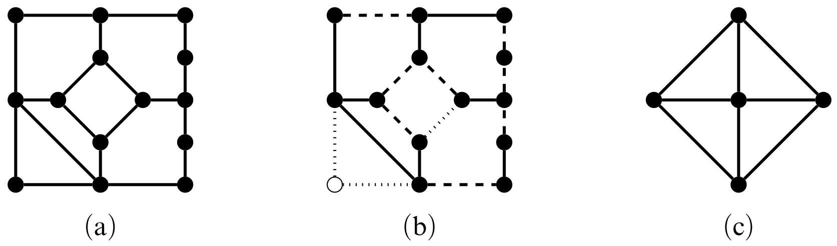

Figure 1.2. The graph in (c) is a minor of the graph in (a) obtained by deleting the dotted edges and the white node and contracting the dashed edges in (b).

of Kuratowski's [82] well-known characterisation of the planar graphs due to Wagner [126], stating that a graph is planar if and only if it excludes K_5 (the complete graph with 5 vertices) and $K_{3,3}$ (the complete bipartite graph with 3 vertices in both parts) as minors. It was a long-standing open question whether a similar characterisation by excluded minors exists for graphs embeddable in other surfaces than the plane or, equivalently, the 2-sphere. Archdeacon [2] gave a list of 35 excluded minors characterising the class of graphs embeddable in the projective plane. No explicit excluded-minor characterisations are known for any surface except the sphere and the projective plane.

However, Robertson and Seymour [108] proved that for every surface such a characterisation exists. Indeed, they proved a much more powerful result known as the Graph Minor Theorem [113]. Let us call a class $\mathcal{C}$ of graphs that is closed under taking minors a *minor ideal*. It is easy to see that for every surface $\boldsymbol{S}$ the class of all graphs embeddable in $\boldsymbol{S}$ is a minor ideal (Figure 9.4 on page 202 illustrates why). There are many other natural graph classes that are minor ideals, for example classes of bounded tree width (see Section 4.1 and Chapter 6), the class of all graphs linklessly embeddable in 3-space (a linkless embedding of a graph G is an embedding where no two cycles of G are linked in the sense of knot theory; see [112] for an explicit excluded-minor characterisation of this class), the class of all graphs knotlessly embeddable in 3-space, the class of all graphs that have a vertex cover of size at most k (a vertex cover of a graph is a set of vertices that contains at least one endvertex of each edge), and the class of all graphs that have a feedback vertex set of size at most k (a feedback vertex set of a graph is a set of vertices that contains at least one vertex of each cycle). Trivially, each minor ideal $\mathcal{M}$ has a characterisation by (possibly infinitely many) excluded minors. The Graph Minor Theorem states that every minor ideal can be characterised by finitely many excluded minors. That is, for every minor ideal $\mathcal{M}$ there is a finite list $H_1, \dots, H_n$ of graphs such that $\mathcal{M}$ is the class of all graphs that do not contain any H_i as a minor.

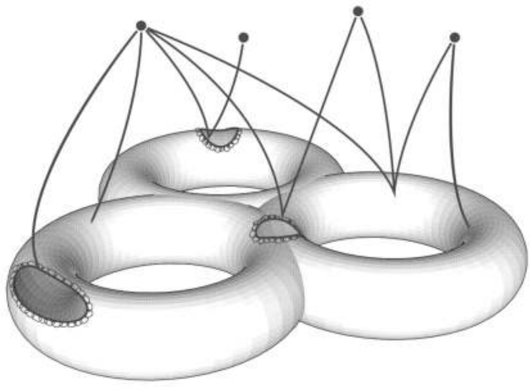

FIGURE 1.3. A graph almost embedded in a triple torus with three vortices and four apices.

To prove the Graph Minor Theorem, in a long series of articles [105] Robertson and Seymour developed a structure theory for graphs with excluded minors. In [111], they proved a structure theorem that says that graphs with excluded minors have a tree decomposition into pieces that are "almost embeddable" into some surface. Intuitively, almost embedding a graph into a surface means first removing a bounded number of vertices from the graph (these vertices are called *apices*) and then drawing the rest of the graph in the surface with no edges crossing except in a bounded number of regions (called *vortices*) in which the surface structure may be violated. The high-level structure is illustrated in Figure 1.3. Each vortex is attached to the boundary of a "hole" in the surface. The vortices may be far from being embeddable in the underlying surface, but they have a fairly simple structure that is controlled by a parameter called the *width* of a vortex. Thus overall there are four parameters in the definition of almost embeddability: the surface, the number of apices, the number of vortices, and the width of the vortices. These parameters are bounded in terms of the excluded minor. (We will give a precise definition of almost embeddability in Chapter 15 and the exact statement of Robertson and Seymour's structure theorem in Chapter 17.)

Besides the Graph Minor Theorem, the structure theorem has found numerous other applications, many of them algorithmic [23, 24, 26, 27, 28, 44, 110]. The structure theorem also plays an important role in this book.

1.2. Treelike decompositions

Tree decompositions and the related notion of tree width have been introduced by Robertson and Seymour in [106]. (Interestingly, several equivalent

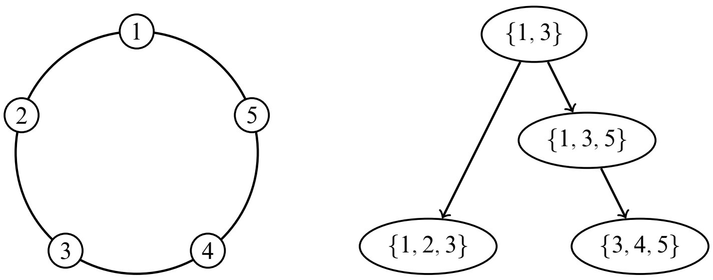

FIGURE 1.4. The cycle C_5 together with a tree decomposition of the cycle.

notions have been introduced independently by other researchers [1, 4, 62, 117].) By now they have developed into a standard tool in structural graph theory and graph algorithms ([104] is a survey). A *tree decomposition* of a graph G consists of a tree T and a mapping β that associates with every node t of T a set $\beta(t)$ of vertices of G subject to certain technical conditions making sure that the structure of the tree T approximates the connectivity structure of G. The set $\beta(t)$ is called the *bag* of the decomposition at t.

Now suppose that we want to define a tree decomposition in some logic. We could try to interpret the tree T in the underlying graph G; that is, define a set of ℓ-tuples of vertices of G representing the nodes of T and define a 2ℓ-ary relation representing the edges of T. Then we could define an $(\ell + 1)$-ary relation to represent the bags. Unfortunately, most interesting tree decompositions are not definable in this way, no matter which logic we use, because the decompositions are not invariant under automorphisms of the graph. What this means is that there may be an automorphism f of G for which we cannot find an automorphism g of T such that for all nodes t we have $\beta(g(t)) = f(\beta(t))$. As an example, consider the decomposition of the cycle C_5 displayed in Figure 1.4. However, only invariant objects are logically definable in the graph.

We resolve this problem by introducing a more general notion of decomposition, which we call *treelike*. In a treelike decomposition, we replace the tree T underlying a tree decomposition by a directed acyclic graph D. The idea is that certain restrictions of D to subtrees yield tree decompositions of G, and by including many such decompositions we can close the treelike decompositions under automorphisms of a graph. To get an impression how treelike decompositions look, consider Figure 1.5, which shows a treelike decomposition of the cycle C_5. The sets displayed in the nodes of the decomposition are the bags. Observe that the four grey nodes form exactly the tree decomposition of C_5 displayed in Figure 1.4. There are many tree decompositions of C_5 contained in the treelike decomposition in a similar way. Note the cyclic structure of the whole decomposition, which reflects the structure of the underlying cycle and

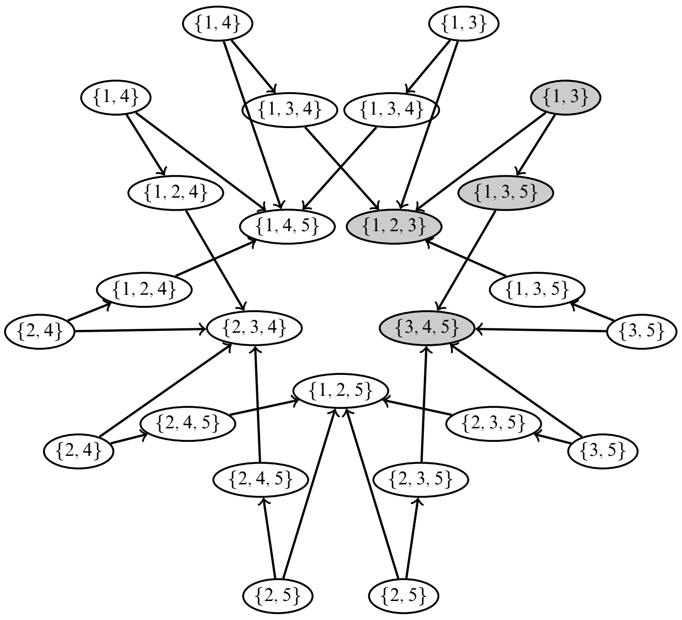

FIGURE 1.5. An automorphism-invariant treelike decomposition of the cycle C_5.

is the reason for the invariance of the decomposition under automorphisms of the cycle. This cyclic structure is lost in a tree decomposition like the one in Figure 1.4.

We extend treelike decompositions to *ordered treelike decompositions*, which one may think of as treelike decompositions together with linear orders of all bags. Our main goal is to prove that certain classes of graphs *admit* IFP-*definable ordered treelike decompositions*. The Definable Structure Theorem for Graphs with Excluded Minors says that this is the case for all classes of graphs excluding some fixed graph as a minor.

1.3. Descriptive complexity theory

Descriptive complexity theory characterises the complexity of computational problems in terms of logical definability. The starting point of the theory was Fagin's Theorem [34] from 1974, stating that existential second-order logic *captures* the complexity class NP. This means that a property of finite structures is decidable in nondeterministic polynomial time if and only if it is definable in existential second-order logic. Similar logical characterisations were later found for most other complexity classes. For example, Immerman [72] and independently Vardi [125] characterised the class PTIME (polynomial time) in terms of least fixed-point logic, and Immerman [74] characterised the classes NL (nondeterministic logarithmic space) and L (logarithmic space) in terms of transitive closure logic and its deterministic variant. However, these logical characterisations of the classes PTIME, NL, and L, and all other known logical characterisations of complexity classes contained in PTIME, have a serious drawback: they only apply to properties of ordered structures, that is, structures with one distinguished relation that is a linear order of the elements of the structure. It is still an open question whether there are logics that characterise these complexity classes on arbitrary, not necessarily ordered, finite structures.

The question of whether there is a logic that captures PTIME was first raised by Chandra and Harel [19] in a fundamental paper on query languages for relational databases. Chandra and Harel asked for a query language expressing precisely those queries that can be evaluated in polynomial time. Gurevich [59] rephrased the question in terms of logic. His precise definition of a "logic capturing PTIME" is subtle; we will discuss it in Chapter 3. Gurevich [60] conjectured that there is no logic capturing PTIME. Note that this conjecture implies PTIME $\neq$ NP, because by Fagin's Theorem there is a logic that captures NP. The question of whether there is a logic capturing PTIME is still open today, and it is viewed as one of the main open problems in finite model theory and database theory. Only partial positive answers are known. To start with, recall the Immerman–Vardi Theorem which states that least fixed-point logic, or equivalently inflationary fixed-point logic IFP, captures PTIME on ordered structures. It is easy to prove that IFP does not capture PTIME on the class of all finite structures. IFP cannot even define the property of a graph having an even number of vertices, but clearly this property is decidable in polynomial time. More generally, IFP "lacks the ability to count". Immerman [73] proposed the extension IFP+C of IFP by *counting operators* as a candidate for a logic capturing PTIME. It was shown by Cai, Fürer, and Immerman in 1992 [16] that IFP+C does not capture PTIME, but it comes surprisingly close. Indeed, Hella, Kolaitis, and Luosto [64] proved that IFP+C captures PTIME on almost all structures (in a precise probabilistic sense).

We shall prove that IFP+C captures PTIME on all classes $\mathcal{C}$ of graphs that admit IFP-definable ordered treelike decompositions. Hence it follows from our Definable Structure Theorem that IFP+C captures PTIME on all classes of graphs excluding some fixed graph as a minor.

1.4. The graph isomorphism problem

It is a long-standing open problem whether there is a polynomial-time algorithm deciding if two graphs are isomorphic. Polynomial time isomorphism tests are known for many natural classes of graphs including the class of planar graphs [67], classes of graphs embeddable in a fixed surface [35, 89], (more generally) classes of graphs with excluded minors [101], and classes of graphs of bounded degree [87]. The isomorphism test for graphs of bounded degree due to Luks [87] involves some nontrivial group theory, and many later isomorphism algorithms build on the group-theoretic techniques developed by Babai, Luks, and others in the early 1980s. In particular, Ponomarenko's [101] isomorphism algorithm for graphs with excluded minors builds heavily on these techniques. It follows from our Definable Structure Theorem that a simple combinatorial algorithm – known as the k-dimensional Weisfeiler–Leman algorithm – decides isomorphism on all classes of graphs excluding some fixed graph as a minor in polynomial time. Here the parameter k of the algorithm depends on the excluded minor.

1.5. The structure of this book

The book has two parts. The first is devoted to the general theory of definable treelike decomposition and its connections with descriptive complexity theory. After two chapters giving the necessary background in graph theory, logic, and descriptive complexity, we introduce in Chapters 4, 5, and 7 (definable, ordered) treelike decompositions and study their basic properties. In Chapter 6 we turn to graphs of bounded tree width and prove that they admit definable treelike decompositions of bounded width. (The *width* of tree decomposition or treelike decomposition is the maximum bag size minus 1, and the *tree width* of a graph is the minimum of the width of all its tree decompositions.) In Chapter 8 we show that every graph has a definable treelike decomposition into its 3-connected components. The first part culminates, in Chapter 9, in a Definable Structure Theorem for Graphs Embeddable in a Surface, stating that for each surface $\boldsymbol{S}$ the class of all graphs embeddable in $\boldsymbol{S}$ admits IFP-definable ordered treelike decompositions.

The first part only uses elementary graph theory. Maybe expanded by additional background material on descriptive complexity theory (for example, [38]), it could be used as the basis of a course on this direction of finite model theory.

The second part is largely devoted to a proof of the Definable Decomposition Theorem for Graphs with Excluded Minors. Instead of going through the chapters one by one, we give an outline of the proof. Actually, we step back to the first part and start with an outline of the proof of the definable decomposition theorems for planar graphs and graphs embeddable in a surface.

Step 1: Planar graphs. We first prove that 3-connected planar graphs admit IFP-definable linear orders. The key step in the proof is to define the facial cycles of a 3-connected planar graph in IFP. We use the fact, going back to Whitney, that the facial cycles (boundaries of the faces) of a 3-connected graph embedded in the plane are precisely the chordless and nonseparating cycles. In particular, this means that the facial cycles are the same for every embedding of the graph. Once we have defined the facial cycles, we can use three parameters to fix one facial cycle and then define in IFP a linear order by "walking around this cycle in spirals".

To show that arbitrary planar graphs admit IFP-definable ordered treelike decompositions, we use the result (proved in Chapter 8) that every graph has an IFP-definable treelike decomposition into its 3-connected components.

Step 2: Graphs embeddable in a surface. We exploit the fact that every surface of positive Euler genus has a noncontractible cycle. Cutting the surface open along such a cycle and glueing disks on the hole(s) yields one or two surfaces of strictly smaller Euler genus.

To define ordered treelike decompositions on graphs embeddable in a surface, we proceed by induction on the Euler genus of the surface. Planar graphs are the base case. In the inductive step, we try to define the facial cycles of a graph embedded in a surface. Either we succeed, then we can use the facial cycles to define a linear order in a similar way as for planar graphs. Or we find a noncontractible cycle along the way. Then we can delete this cycle, apply the induction hypothesis to the resulting graph embeddable in one or two surfaces of strictly smaller Euler genus, and extend the decomposition to the original graph.

Step 3: Almost planar graphs. Remember our informal description of almost-embeddable graphs. Let $\mathcal{A}(p,q,r,s)$ be the class of all graphs almost embeddable in a surface of Euler genus at most r with at most s apices and at most q vortices, each of width at most p. In this and the following step, we want to prove that for all p,q,r,s the class $\mathcal{A}(p,q,r,s)$ admits IFP-definable ordered treelike decompositions. We proceed by induction on $q+r$. We already know how to deal with the class $\mathcal{A}(0,0,0,0)$ of planar graphs and, more generally, the class $\mathcal{A}(0,0,r,0)$ of all graphs embeddable in a surface of Euler genus at most r.

In this step, we consider the classes $\mathcal{A}(p,1,0,0)$ of *almost planar graphs*. We think of almost planar graphs as graphs being embedded into a disk with a vortex glued on the boundary of the disk. Figure 1.6 shows an example. In the key lemma of this step, and actually one of the most difficult lemmas of the whole book, we show that the facial cycles of an almost planar graph that are sufficiently far from the vortex do not depend on the specific embedding. That is, no matter how we divide the graph into a vortex and a part embedded in the disk, these cycles will end up in the disk, and they will be facial cycles. Moreover, these cycles are IFP-definable. We call the subgraph of the graph induced by these cycles the *centre* of the graph. Using the facial cycles, we can define a linear order on each connected component of the centre. If we contract each connected component of the centre to a single vertex, we obtain a graph of tree width bounded by $O(p^2)$, which we call the *skeleton* of our original

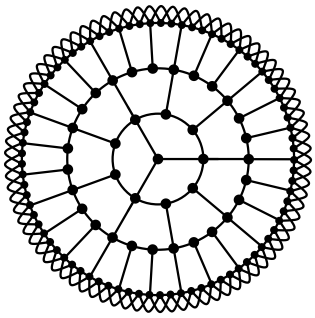

FIGURE 1.6. An almost planar graph (the outer edges connect vertices of distance two on the cycle).

graph. We define a treelike decomposition of bounded width of the skeleton, and we can extend the decomposition to an ordered treelike decomposition of the original graph using the linear orders of the components of the centre.

Step 4: Almost-embeddable graphs. We prove that the classes $\mathcal{A}(p, q, r, s)$ admit IFP-definable ordered treelike decompositions by an inductive construction similar to, but far more complicated than, the one in Step 2.

Step 5: Graphs with excluded minors. Let $\mathcal{C}$ be a class of graphs excluding some fixed graph as a minor. By Robertson and Seymour's structure theorem, there are p, q, r, s such that all graphs in $\mathcal{C}$ have a tree decomposition into pieces from $\mathcal{A}(p, q, r, s)$. If we could define such a decomposition in IFP, then we could use the definable decomposition of the graphs in $\mathcal{A}(p, q, r, s)$ obtained in the previous step to define ordered treelike decompositions of the graphs in $\mathcal{C}$. But unfortunately I do not know how to define such a decomposition in IFP.

Instead, we repeatedly apply the construction of the previous steps to inductively build up an ordered treelike decomposition of a graph in $\mathcal{C}$ from partial decompositions in a bottom-up fashion. To be able to do this, we have to prove generalisations of the results from the previous steps, so-called *completion lemmas*, which roughly say that if we already have ordered treelike decompositions of parts of a graph, and if the part of the graph that is not covered by these partial decompositions has a nice structure, such as being almost embeddable in some surface, then we can complete the partial decompositions to an ordered treelike decomposition of the whole graph. The formal framework of *pre-decompositions* and *completions* will be introduced in Chapter 12.

Another difficulty of the proof is that we need the graphs we decompose to be not only 3-connected, but *quasi*-4-*connected*. Quasi-4-connectedness is a new notion introduced in Chapter 10. We prove that all graphs have definable decompositions into their "quasi-4-connected components." Unfortunately, these decompositions turn out to be quite complicated. As an immediate