National and Kapodistrian University of Athens

Graduate Program in Logic, Algorithms
and Computation

MSc thesis

On the computability of obstruction sets for well–quasi–ordered graph classes

Iosif Salem

supervised by:
Dimitrios M. Thilikos

September 17, 2012

To the memory of my father,
Guido Salem

Η παρούσα Διπλωματική Εργασία

εκπονήθηκε στα πλαίσια των σπουδών

για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδίκευσης

στη

Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

που απονέμει το

Τμήμα Μαθηματικών

του

Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 17η Σεπτεμβρίου 2012 από Εξεταστική Επιτροπή

αποτελούμενη από τους:

Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1. : Δ. Θηλυκός	Αναπλ. Καθηγητής	……
2. : Ε. Κυρούσης	Καθηγητής	……
3. : Στ. Κολλιόπουλος	Αναπλ. Καθηγητής	……

Abstract

In this MSc thesis we are going to present algorithms for computing obstruction sets of well–quasi–ordered graph classes. Neil Robertson and Paul Seymour's Graph Minor Theorem (GMT) [27] guarantees that any minor–closed graph class has a finite obstruction set. If $\mathcal{C}$ is such a class, the obstruction set of $\mathcal{C}$, $\mathbf{obs}_m(\mathcal{C})$, is the minimal set of graphs $\mathcal{H}$ such that G belongs to $\mathcal{C}$ if and only if none of the graphs in $\mathcal{H}$ is contained as a minor in G. The analogous result for another well–quasi–ordering, the immersion ordering, was shown in the same series of papers (Graph Minors), in [30]. But this results are non-constructive; we know that a minor or immersion–closed graph class has a finite obstruction set but the GMT does not imply any algorithm for computing it. K. Cattell, M. J. Dinneen, R. Downey, M. R. Fellows and M. Langston in "On computing graph minor obstruction sets" [6] and I. Adler, M. Grohe and S. Kreutzer in "Computing Excluded Minors" [1] present algorithms to overcome this problem for minor–closed graph classes, as well as, applications of their methods proving that the obstruction sets of various graph classes are computable, such as the union problem. By adapting some of the methods of [1] to immersions, the analogue result for immersion obstruction sets and an algorithm for the union problem on immersion–closed graph classes are proven in [17] by A. Giannopoulou, D. Zoros and the author, under the supervision of D. M. Thilikos.

Keywords: Graph Minor Theorem, obstruction set, Monadic Second-Order Logic, immersion, tree-width.

Acknowledgements

This master thesis would not have been possible without the support of many people. I would like to thank my supervisor Dimitrios M. Thilikos for introducing me to a very interesting theory that combines Algorithmic Graph Theory and Mathematical Logic, and also for introducing me to Theoretical Computer Science through his lectures in Theory of Algorithms, Computational Complexity, Recursion Theory and Graph Theory during my undergraduate studies in the Department of Mathematics of the University of Athens. In addition, I would like to thank Archontia Giannopoulou and Dimitris Zoros for collaborating with me and for making this project even more interesting.

I would also like to thank the members of the supervisory committee Stavros Kolliopoulos and Lefteris Kirousis, and all the professors of the graduate program in "Logic, Algorithms and Computation" (MPLA) for their stimulating lectures and for giving a model of how a professor should be like. I should also thank the secretariats of MPLA for being so helpfull, patient and efficient anytime needed.

Above all, I would like to thank my parents Guido and Sarra and my sister Louiza for all their support and love, and for many things that words cannot express. I must also thank my close family and some family friends, such as Zak Koune, who has been always there to help and guide, even before I realize that I need either of them.

Moreover, I thank Nikos Cheilakos, Marilena Karanika, Babis Kontos, Katerina Kosta, Tina Miligkou, Yiannis Nikolakopoulos, Theodora Ntoka, Antonis Panagopoulos, Kanellos Papantonopoulos, Katerina Samari, Haris Tampakopoulos, Eirik George Tsarpalis and Dimitra Vouterou for being friends in good and hard times.

Finally, I thank all the fellow students of the Laboratory of the Department of Mathematics (UoA) and also the co-producers of *Lab Radio* (`http://pclab.math.uoa.gr/labradio`) for being a pool of creative and interesting people to interact with. Of course, for this environment I should thank prof. Evangelos Raptis. I should not forget to thank Aggeliki and Hara for making me see lunch in the University's restaurant from another perspective.

Contents

Introduction

A classic result by Wagner (equivalent to the well known Kuratowski's theorem[1] for planar graphs) states that a graph is planar if and only if it does not contain as a minor K_5 or $K_{3,3}$. So the algorithm that, given a graph G answers *yes* if no graph in $\{K_5, K_{3,3}\}$ is contained in G as a minor and *no* otherwise, is an algorithm that *decides* the minor–closed graph class of planar graphs. A first question on the above algorithm would be: do we have a subroutine to check if a graph is contained into another as a minor? The answer is yes! Neil Robertson and Paul Seymour showed in [26] that for a known graph H, given a graph G we can check in cubic time ($O(|V(G)|^3)$) if H is a minor of G. So our algorithm decides if a graph is planar in cubic time. Of course, planarity testing can be done in linear time ($O(|V(G)|)$) by the Hopcroft–Tarjan algorithm [19] (see also [21]), but we use this familiar problem to describe the framework that we want to work on. Thus, on a second look one would wonder, can we generalize this algorithm to other graph classes and other relations on graphs?

Lets step back and look at the framework of the algorithm. We have a minor–closed graph class, call it $\mathcal{C}_p$, and a set of graphs for that class $\mathcal{O}_{\mathcal{C}_p} = \{K_5, K_{3,3}\}$. We describe the algorithm in pseuodcode as shown in Algorithm 1. On input G the algorithm checks if none of the *forbidden graphs* is contained in G, and if this holds then it accepts and in any other case it rejects the input. Since we know algoritms [26, 30] to check if a fixed graph H is a minor, or immersion of a graph G in cubic time, the only ingredient missing is the set $\mathcal{O}$ of forbidden graphs, or even better the mininal set of forbidden graphs, which we call the **obstruction set** for a graph class.

Algorithm 1 Algorithm for the minor-closed graph class $\mathcal{C}_p$

```
Input: a graph G
for every H ∈ O_{C_p} do
  if H is contained as a minor in G then
    return "no"
  end if
end for
return "yes"
```

[1] As shown independently by Kuratowski and Pontryagin, a graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$. The result was then proved for minors by Wagner and it is easy to see that the two theorems are equivalent.

Klaus Wagner conjectured[2] that *for every infinite set of finite graphs, one of its members is a minor of another* i.e. the minor relation on finite graphs is a well–quasi–order (WQO). And that (Robertson and Seymour theorem, Graph Minor Theorem, GMT, [27], 2004) was proved in a series of papers by Robertson and Seymour, called the Graph Minors (1983–2010). Later on, in [30] the immersion relation was also proven to be a well–quasi–order. Consequently, every minor or immersion-closed graph class has a **finite** obstruction set. Thus, we have the last ingredient to generalize Algorithm 1 but are we done? Unfortunately, the answer here is no! And it is negative because the GMT guarantees the *existense* of a finite obstruction set, but since the proof is non-constructive [23], it gives us no algorithm to construct the obstruction set.

The natural question that arises is

> "*what pieces of information do we need for a minor or immersion-closed graph class to transform the existense of the obstruction set to a construction of it?*"

M.R. Fellows and M. Langston proved in [16] that an algorithm for deciding membership in a minor-closed graph class $\mathcal{F}$ is not sufficient to construct the obstruction set of $\mathcal{F}$. However, the results of "*On computing graph minor obstruction sets*" by K. Cattell, M.J. Dinnen, R.G. Downey, M.R. Fellows and M.A. Langston, [6], and "*Computing Excluded Minors*" by I. Adler, M. Grohe and S. Kreutzer, [1], showed that there are algorithms that compute obstruction sets of minor-closed graph classes, but for that we need more information than just an algorithm for membership in $\mathcal{F}$.

The approach of Cattell et al. is an algorithmic one. The authors employ an analogue of the Myhill–Nerode theorem [15] and define congruences based on a minor–closed class that split the class in a finite number of equivalence classes, of which they find the mininal elements. They demand the knwoledge of a bound on the treewidth of the obstructions but another algorithm for obstruction set computation is also proven, demanding other information about the congruence but not the bound on the treewidth. The approach of Adler et al. requires that the minor–closed graph class we examine is definable in layerwise monadic second–order logic (layerwise MSO). Again a bound on the treewidth of the obstructions is needed and Seese's theorem [31] plays an important role. We are going to present these two approaches in chapter 3 and 4 respectively.

A main application given in both of the papers is the computation of the obstruction set of $\mathcal{F}_1 \cup \mathcal{F}_2$ given the obstruction set of two minor-closed graph classes, $\mathcal{F}_1$ and $\mathcal{F}_2$. Using some of the machinery of [1], A. Giannopoulou, D. Zoros and the author, under the supervision of our advisor D.M. Thilikos, proved that there is an algorithm to compute the obstruction set of $\mathcal{F}_1 \cup \mathcal{F}_2$ given the obstruction set of two immersion–closed graph classes, $\mathcal{F}_1$ and $\mathcal{F}_2$. This work was published on SWAT 2012 and is presented in chapter 5, in the scope of computability, as an application of the previous chapters. Chapter 5 ends with some concluding remarks, and of course all the basic notions and preliminaries are presented in chapter 2.

[2] The conjecture is known to be Wanger's, but Wagner has said that he never made this conjecture! [12]

But have we solved the problem of constructibility of the obstruction sets? The answer is almost yes. And that is, because eventhough we have algorithms to compute the obstruction sets of minor and immersion-closed graph classes, the running time of these algorithms is not constructive. This means that we know that our algorithms will stop having produced the obstruction set, but we don't know when. This problem is still open and we are not sure if we can bound the running time of the algorithms or not.

Chapter 1

Preliminaries

1.1 Basics

Lets begin with the definitions of some basic notions. $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of natural numbers, integers and real numbers respectively. By $[n]$ we denote the set $\{1, \ldots, n\}$ of the first n integers. A (simple) graph G is defined to be an ordered pair (V, E) consisting of two sets: the set of vertices (or nodes) V or $V(G)$ and the set of edges E or $E(G)$, where an edge is a two-element set including elements from V: $E \in \{X \subset \mathcal{P}(V) \,|\, \forall e \in X : \ |e| = 2\}$. Schematically, we draw graphs by representing the elements of V as (labeled) nodes and the edges with lines between the two corresponding nodes (endpoints). In figure 1.1 we can see some examples of graphs.

A directed graph is a simple graph where the edges are not sets but ordered pairs, i.e. elements of $V \times V$, and a weighted graph is a simple graph where a real number (the weight) is assigned to each edge. We are going to deal with graphs that are undirected, unweighted and contain no loops (edges with the same endpoints) or multiple edges between two vertices (E is a set, so an edge $\{v_1, v_2\}$ can connect two vertices at most once).

In a graph $G = (V, E)$ the degree of a vertex v is the number of its neighbours. We write $deg_G(v) = |\{w \in V \,|\, \{v, w\} \in E\}|$. The maximum degree of $G = (V, E)$, is $\Delta(G) = max\{deg_G(v) \,|\, v \in V\}$ and its line graph is the graph $L(G) = (E, X)$, where $X = \{\{e_1, e_2\} \,|\, e_1, e_2 \in E \wedge e_1 \cap e_2 \neq \emptyset \wedge e_1 \neq e_2\}$. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomporphic if there is a bijection[1] f from V_1 to V_2, such that $\{v, w\} \in E_1 \Leftrightarrow \{f(v), f(w)\} \in E_2$. For example try to see that all graphs in figure 1.1 are isomporphic[2]. Given two graphs G and H, the *lexicographic product* $G \times H$, is the graph with $V(G \times H) = V(G) \times V(H)$ and $E(G \times H) = \{\{(x, y), (x', y')\} \mid (\{x, x'\} \in E(G)) \vee (x = x' \wedge \{y, y'\} \in E(H))\}$.

Now lets continue with relations on graphs. We say that a graph $G_1 = (V_1, E_1)$ is a subgraph of a graph $G_2 = (V_2, E_2)$, $G_1 \subseteq G_2$, if a graph isomorphic to G_1 can be

[1] A one-to-one and onto function.

[2] The first graph of figure 1.1 is called the Petersen graph. Eventhough all graphs in figure 1.1 are isomorphic its first draw is more popular.

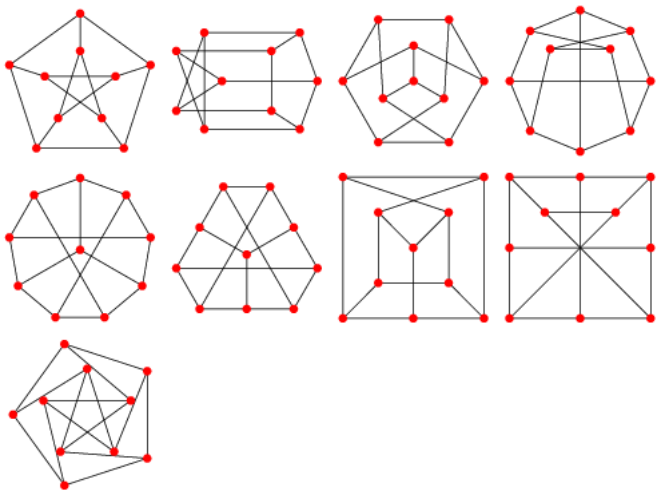

Figure 1.1: Some graphs.

obtained by G_2 after a sequence of vertex or edge deletions. The operations of vertex and edge deletion are defined as follows: for $v \in V(G), e \in E(G)$, $G\backslash v = (V(G) \setminus \{v\}, E')$, where E' is obtained by $E(G)$ after removing all the edges that have an endpoint in v, and $G\backslash e = (V(G), E(G) \setminus \{e\})$. Given an edge $e = \{x, y\}$ of a graph G, the graph $G\backslash e$ is obtained from G by contracting the edge e, that is, the endpoints x and y are replaced by a new vertex v_{xy} which is adjacent to the old neighbors of x and y (except x and y). A graph G_1 is a *topological minor* of a graph G_2, $G_1 \leq_{tp} G_2$, if a graph isomorphic to G_1 can be obtained by a subdivision of a subgraph of G_2. A graph H is a *subdivision* of a graph G, if G is obtained from H by replacing some (or maybe none) edges of H with paths.

A graph H is a *subdivision* of a graph G if a graph isomorphic to G can be obtained by H by replacing some of the edges of H with paths. Furthermore, we say that H is a *topological minor* of G, $H \leq_{tp} G$, if H is a subdivision of a subgraph of G. We are now ready to define the two relations on graphs that are basic for our study.

A graph H is a *minor* of G, $H \leq_m G$, if an isomorphic graph to H can be obtained from a subgraph of G after a sequence of edge contractions. An equivalent definition (which will be useful in chapter 3) would state that $H \leq_m G$ if there is a function that maps every vertex v of H to a connected subgraph $B_v \subseteq V(G)$, such that for every two distinct vertices v, w of H, B_v and B_w share no common vertex, and for every edge $\{u, v\}$ of H, there is an edge in G with one endpoint in B_v and one in B_u.

We say that H is an *immersion* of G (or H is immersed in G), $H \leq_{im} G$, if there is an injective mapping $f : V(H) \to V(G)$ such that, for every edge $\{u, v\}$ of H, there is a path from $f(u)$ to $f(v)$ in G and for any two disjoint edges of H the corresponding paths in G are edge-disjoint, that is, they do not share common edges[3]. Furthermore,

[3]There is an equivalent definition of immersion stating that $H \leq_{im} G$, if a graph isomorphic to H can be obtained from a subgraph of G after a sequence of edge–liftings. The operation of edge–lifting is

two paths are called vertex-disjoint if they do not share common vertices. Naturally, the above relations on graphs also define orderings on graphs (e.g. minor ordering, immersion ordering).

1.2 WQO, GMT and obstructions

Definition 1.1. *A well-quasi-ordering, WQO, on a set $\mathcal{C}$ is a quasi-ordering (a reflexive and transitive binary relation) which has no infinite descending sequences and no infinite antichains.*

An infinite antichain in $\mathcal{C}$ is an infinite sequence of elements in $\mathcal{C}$ such that for every $x_i, x_j \in \mathcal{C}$, with $i < j$, we have that $x_i \not\leq x_j$. For example, we know by Kruskal's Tree Theorem [22] that the set of rooted trees is WQO by the topological minor relation.

Theorem 1.2 (Graph Minor Theorem, [27])**.** *The class of finite graphs is well-quasi-ordered by the minor relation.*

Robertson and Seymour in the last of the Graph Minors series also proved the following:

Theorem 1.3 ([30])**.** *The class of finite graphs is well-quasi-ordered by the immersion relation.*

Definition 1.4. *Let $\leq$ be a relation on graphs, such as $\leq_m$ or $\leq_{im}$. A graph class $\mathcal{C}$ is $\leq$–closed, or closed under $\leq$, if for every graph G in $\mathcal{C}$, if $H \leq G$ for a graph H, then H also belongs to $\mathcal{C}$.*

As an example, consider the class of planar graphs, $\mathcal{C}_p$. If G is planar and $H \leq_m G$, then H is also planar, since vertex and edge deletion and edge contraction preserve planarity. So $\mathcal{C}_p$ is a minor–closed graph class. Since edge lifting also preserves planarity, $\mathcal{C}_p$ is also an immersion-closed graph class.

Definition 1.5. *The obstruction set of a $\leq$–closed graph class $\mathcal{C}$, $\mathbf{obs}(\mathcal{C})$, is the minimal set of graphs $\mathcal{H}$ satisfying the following property:*

$$G \in \mathcal{C} \Leftrightarrow \text{for every } H \in \mathcal{H},\ H \not\leq G$$

As a corollary of theorems 1.2 and 1.3, we get the following:

Corollary 1.6. *For every minor or immersion–closed graph class $\mathcal{C}$, the set $\mathbf{obs}(\mathcal{C})$ is finite.*

It is obvious that if a graph class $\mathcal{C}$ is minor–closed and also immersion–closed, the minor obstructions $\mathbf{obs}_m(\mathcal{C})$ are not the same with the immersion obstructions $\mathbf{obs}_{im}(\mathcal{C})$, except for some trivial cases. In addition, obstruction sets are often large sets of graphs. As an example we give the following theorem (we will not define tree–depth since this theorem is given only as an example of the cardinality of an obstruction set).

---

the replacement of two edges $\{v, x\}, \{x, w\}$ with a new one $\{v, w\}$.

Theorem 1.7 (A. Giannopoulou and D.M. Thilikos, [18]). *Let $\mathcal{F}$ be the ($\leq_m$–closed) class of all graphs with tree-depth at most k. Then $|\mathcal{O}| \geq 2^{2^k-(2k+1)} + 2^{2^k-(k+1)}$, where $\mathcal{O}$ is the obstruction set of the class.*

1.3 Treewidth

This section is about a parameter on graphs called treewidth, introduced by N. Robertson and P. Seymour in [25]. Intuitively this parameter indicates how much a graph resembles a tree. For a formal approach we have to define tree decompositions.

A *tree decomposition* of a graph G is a pair (T, B), where T is a tree and B is a function that maps every vertex $v \in V(T)$ to a subset B_v of $V(G)$ such that:

(i) for every edge e of G there exists a vertex t in T such that $e \subseteq B_t$,

(ii) for every $v \in V(G)$, if $r, s \in V(T)$ and $v \in B_r \cap B_s$, then for every vertex t on the unique path between r and s in T, $v \in B_t$ and

(iii) $\bigcup_{v \in V(T)} B_v = V(G)$.

We usually treat a node t of T and the corresponding set of vertices $B(t)$ as equal. So when we refer to the vertices in t of T it will be clear that we mean $B(t)$.

The *width* of a tree decomposition (T, B) is $\text{width}(T, B) := \max\{|B_v| - 1 \mid v \in V(T)\}$ and the *treewidth* of a graph G, $\mathbf{tw}(G)$, is the minimum over the $\text{width}(T, B)$, where (T, B) is a tree decomposition of G.

If on the above definition we restrict T to be a path, we define *path decomposition* and *pathwidth*, $\mathbf{pw}(G)$.

A graph class $\mathcal{C}$ is said to have *bounded treewidth* if there is a $k \in \mathbb{N}$, such that for every graph G in $\mathcal{C}$, $\mathbf{tw}(G) \leq k$.

Despite the fact that treewidth will play a key role in both approaches studied for obstruction set computation, it is a parameter with a wide impact on theoretical computer science and therefore well studied. Many NP-hard problems on graphs become polynomial time solvable when we know a bound on the treewidth, with exact algorithms or approximate. For example, computing the maximum independent set[4] of a graph is a classic NP-complete problem and it is also hard to approximate.

Theorem 1.8 (Arnborg and Proskurowski [2]). *The size of the largest independent set in a graph of treewidth k can be computed in $O(4^k k^2 n)$ time.*

In both approaches studied to compute the obsrtuction set of a graph class $\mathcal{C}$ we are going to require for $\mathbf{obs}(\mathcal{C})$ to have bounded treewidth. This bound always exists, since the obstruction sets of minor or immersion–closed graph classes are finite. On the other hand, there are many graph classes that have themselves bounded treewith and

[4] An independent set of a graph G is a subset X of its vertices, such that no two vertices in X are adjacent in G.

as an example we give the following theorem (for a proof of the theorem and a study on treewidth see [14]).

Theorem 1.9. *A minor-closed family of graphs has bounded treewidth if and only if it excludes at least one planar graph.*

On the contrary, there are many classes with no bounded treewidth. A simple example is, again, the minor–closed class of planar graphs, $\mathcal{C}_p$. All the grid graphs ($P_n \times P_m$, $n, m \in \mathbb{N}$, $\times$ is the graph Cartesian product and P_i is a path with i nodes) are included in $\mathcal{C}_p$, but since $\mathbf{tw}(P_n \times P_n) = n$, for any $n \in \mathbb{N}$ we can find a graph in $\mathcal{C}_p$ with treewidth exactly n. For a survey on treewidth see [5].

Chapter 2

An algorithmic approach

In [6] K. Cattell, M. Dinneen, R. Downey, M. Fellows and M. Langston present algorithms for obstruction set computation of minor–closed graph classes, which they call *ideals*, remarking that their methods can be adapted to other partial–orders on graphs such as immersions and topological orders. The first algorithm appeared in [15] (but its correctness was fully proved for the first time in [6]) and is based on the Myhill–Nerode theorem for regular languages. However, an upper bound of the treewidth of the obstructions is an essential ingredient of the algorithm and this piece of information is not always known. Thus, the authors introduce the notion of second–order congruence for a graph property, which allows them to compute the obstruction set of an ideal requiring other information (about the congruence) but not the upper bound of the treewidth of the obstructions. In this chapter we present these algorithms.

2.1 Introduction

The first algorithm is an analogue of the Myhill–Nerode theorem ([15]), but before recalling the theorem it is necessary to define some relevant notions. Let $L \subseteq \Sigma^*$ be a language and $\sim$ an equivalence relation. The canonical equivalence relation $\sim_L$ is, $x \sim_L y$ if and only if, $\forall z \in \Sigma^*$, $xz \in L \iff yz \in L$. We say that $\sim$ is a *congruence* if, $\forall z \in \Sigma^*$, $x \sim y \iff xz \sim yz$ and $zx \sim zy$. We can now state the Myhill–Nerode theorem.

Theorem 2.1 (Myhill–Nerode)**.** *Let $L \subseteq \Sigma^*$. If an equivalence relation $\sim$ that is decidable with a known algorithm satisfies the following:*

(1) $\sim$ has finite index

(2) $\sim$ is a congruence with respect to concatenation

(3) $x \sim y \implies x \sim_L y$,

then L is regular and an algorithm is known to compute a DFA[1] that recognizes L.

[1] Deterministic Finite Automaton.

Instead of words and languages over an alphabet we deal with graphs and graph classes. To define an operation analogous to concatenation[2] we have to define t-boundaried graphs in order to "glue" graphs together (on the boundary), as concatenation does with words.

Definition 2.2. *A t-boundaried graph $G = (V, E, B, f)$ is a simple graph $G = (V, E)$ together with:*

(i) a subset of vertices $B \subseteq V$, which we call the boundary, such that $|B| = t$

(ii) a bijection $f : B \to \{1, \ldots, t\}$ (a labeling of the boundary vertices)

We denote by $\mathcal{U}^t_{large}$ the set of all t–boundaried graphs and by $\underline{G} = (V, E)$ the *underlying graph* of the t–boundaried graph $G = (V, E, B, f)$. For simplicity, sometimes we will not distinguish between the t–boundaried graph and the the underlying graph. So for example when we write $G \in \mathcal{F}$ and G is a t–boundaried graph, we actually mean $\underline{G} \in \mathcal{F}$. We now define the following operation on t–boundaried graphs.

Definition 2.3. *Let $G_1 = (V_1, E_1, B_1, f_1)$ and $G_2 = (V_2, E_2, B_2, f_2)$ be two graphs in $\mathcal{U}^t_{large}$, then $G_1 \oplus G_2$ is the t-boundaried graph, obtained by the disjoint union of the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, by identifying each vertex $u_1 \in B_1$ with the vertex $u_2 \in B_2$ for which $f_1(u_1) = f_2(u_2)$.*

To explain the above, the operation of $G_1 \oplus G_2$, where $G_1, G_2 \in \mathcal{U}^t_{large}$, produces a new t–boundaried graph which is the disjoint union of the two underlying graphs, having the boundary vertices with the same labels identified.

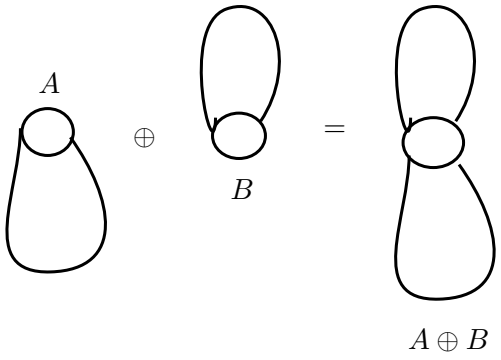

Figure 2.1: $A \oplus B$

Definition 2.4. *Let $\mathcal{F}$ be a graph class and $X, Y \in \mathcal{U}^t_{large}$. The large canonical congruence for $\mathcal{F}$ is defined on $\mathcal{U}^t_{large}$ as follows: $X \approx_{\mathcal{F}} Y$ if and only if,*

$$\forall Z \in \mathcal{U}^t_{large} : (X \oplus Z \in \mathcal{F}) \iff (Y \oplus Z \in \mathcal{F})$$

[2]If v, w are two words (strings) of Σ^*, the concatenation of v and w produces the word vw–we just join the two strings into a new one.

Definition 2.5. *The small treewidth universe, $\mathcal{U}^t_{tree}$, is the set of all t–boundaried graphs having tree decomposition of width $t-1$ (equivalently, treewidth[3] at most t) for which the set of boundary vertices is the set of vertices indexed by the root of the tree.*

The small pathwidth universe, $\mathcal{U}^t_{path}$, is the set of all t–boundaried graphs having a path decomposition of width $t-1$ (equivalently, pathwidth[4] at most t) for which the set of boundary vertices is the last set of the decomposition.

When our statements hold for both $\mathcal{U}^t_{tree}$ and $\mathcal{U}^t_{path}$ we will simply write $\mathcal{U}^t_{small}$. We remark that if G is a graph in $\mathcal{U}^t_{tree}$ (resp. $\mathcal{U}^t_{path}$) the trees (resp. paths) of a minimal tree (resp. path) decomposition of G are rooted and the root of the tree (resp. end of the path) has size exactly t. The following simple lemma will be useful in the sequel.

Lemma 2.6. *Let $A, B \in \mathcal{U}^t_{small}$. Then $\mathbf{tw}(A \oplus B) \leq t$ (equiv. $\mathbf{pw}(A \oplus B) \leq t$).*

Proof. Let $A, B \in \mathcal{U}^t_{tree}$ (the same arguements hold for the case of $\mathcal{U}^t_{path}$). We consider a tree decomposition (T_1, B_1) of A and a tree decomposition (T_2, B_2) of B such that both decompotitions have width $t-1$ and the set of boundary vertices is the set of vertices indexed by the root of every tree. We concider the tree T which occurs from the disjoint union of the trees T_1 and T_2 by identifying their roots, which is also the root of T, and we name the boundary vertices of A and B with their labels in $\{1, \ldots, t\}$. We also consider the function B which is the union of B'_1 and B'_2, where B'_i is the function B_i having replaced any occurence of a boundary vertex with its label. Then it is easy to check that (T, B) is a tree decomposition of $A \oplus B$ and therefore $\mathbf{tw}(A \oplus B) \leq t$ (in the case of $\mathcal{U}^t_{path}$ we would join the ends of the paths of the path decompositions of A and B). □

Similarly with $\approx_{\mathcal{F}}$ (definition 2.4) we define the small canonical congruence $\sim_{\mathcal{F}}$ in $\mathcal{U}^t_{small}$.

Definition 2.7. *Let $X, Y \in \mathcal{U}^t_{small}$. The small canonical congruence is defined by $X \sim_{\mathcal{F}} Y$ if and only if*

$$\forall Z \in \mathcal{U}^t_{small} : (X \oplus Z \in \mathcal{F}) \iff (Y \oplus Z \in \mathcal{F})$$

We denote by $[x]_\sim$ the equivalence class of x with respect to $\sim$ and we say that $\sim$ has *finite index* on $\mathcal{U}$, if $\sim$ has a finite number of equivalence classes in $\mathcal{U}$. We say that $\sim$ *refines* $\approx$, where $\sim$ and $\approx$ are two equivalence relations in $\mathcal{U}$, if

$$\forall x, y \in \mathcal{U} : x \sim y \implies x \approx y$$

If we consider $\sim_{\mathcal{F}}$ and $\approx_{\mathcal{F}}$ it is trivial to see that $\approx_{\mathcal{F}}$ refines $\sim_{\mathcal{F}}$ (the two relations though, might not coincide). Finally, it is shown in [11] that on $\mathcal{U}^t_{tree}$ the large canonical congruence $\approx_{\mathcal{F}}$ has finite index if and only if the small canonical congruence $\sim_{\mathcal{F}}$ has finte index, for a graph class $\mathcal{F}$.

[3]For a t-boundaried graph G, $\mathbf{tw}(G) = \mathbf{tw}(\underline{G})$.

[4]As above, for $G \in \mathcal{U}^t_{large}$, $\mathbf{pw}(G) = \mathbf{pw}(\underline{G})$.

2.2 An algorithm for $\mathbf{obs}(\mathcal{F})$

After defining the analogous notions of the Myhill–Nerode theorem (2.1) for graphs, we can now state an analogue of the theorem, which is the first algorithm in our study for obstruction set computation.

Theorem 2.8 ([6])**.** *Let $\mathcal{F}$ be a minor closed graph class. If we have the following three pieces of information:*

(i) An algorithm to decide membership in $\mathcal{F}$ (of any time complexity).

(ii) A bound B on the maximum treewidth of the obstructions for $\mathcal{F}$.

(iii) For $t = 1, \ldots, B+1$ a decision algorithm for a finite–index congruence $\sim$ on t–boundaried graphs that refines the small canonical congruence for $\mathcal{F}$.

Then we can effectively compute the obstruction set $\mathcal{O}$ for $\mathcal{F}$.

Proof. In short, the information given are enough to bound the search space of the obstructions and by using the algorithm for $\mathcal{F}$–membership we find the minimal graphs (with respect to an ordering that will be defined) that do not belong to $\mathcal{F}$, i.e. the obstructions of $\mathcal{F}$. We can now state the algorithm and then explain it and prove its correctness.

Algorithm 2 $\mathbf{obs}(\mathcal{F})$ computation

for $t = 1, \ldots, B+1$ **do**
  $j \leftarrow 1$
  while a stop signal is not detected **do**
    generate all graphs of the j^{th} generation and store them in $\mathcal{G}_t$
    $j \leftarrow j + 1$
  end while
  store all the $\leq$–mininal graphs of $\mathcal{G}_t$ in $\mathcal{M}_t$
  for every $G \in \mathcal{M}_t$ **do**
    if $\underline{G} \notin \mathcal{F}$ then store $\underline{G}$ in $\mathcal{O}_t$
  end for
end for
return $\bigcup_{t=0}^{B+1} \mathcal{O}_t$

Algorithm 2 has two loops. The outer loop $t = 1, \ldots, B+1$ specifies that we are currently working in $\mathcal{U}^t_{tree}$. In the inner loop the graphs of $\mathcal{U}^t_{tree}$ are generated, until a stop signal is detected and all these graphs are stored in $\mathcal{G}_t$. When this process ends, we store all the $\leq$–minimal graphs of $\mathcal{G}_t$ in $\mathcal{M}_t$ ($\leq$ will be defined in the sequel) and this process is completed in a finite number of steps, since $\mathcal{G}_t$ is finite. Then, by using the algorithm for $\mathcal{F}$–membership we discard all the underlying graphs of $\mathcal{M}_t$ that

belong to $\mathcal{F}$ and store the remaining elements of $\mathcal{M}_t$ in $\mathcal{O}_t$. Since $\mathcal{O}_t$ contains all the $\leq$–minimal graphs of $\mathcal{U}^t_{tree}$ that do not belong in $\mathcal{F}$, the algorithm ends by returning $\bigcup_{t=0}^{B+1} \mathcal{O}_t$, which is (as we will prove in the sequel) the obstruction set of $\mathcal{F}$ (the obstruction set of a class $\mathcal{F}$ can be also defined as the set of all $\leq$–minimal graphs of the complement of $\mathcal{F}$).

To complete the description of the algorithm we need to explain (a) how the graphs of the inner loop are generated (what is the j^{th} generation in Algorithm 2), (b) the $\leq$ ordering, and (c) the stop signal.

For (a) we define $size(X)$, where $X \in \mathcal{U}^t_{tree}$, to be the number of nodes in the smallest possible rooted tree of a tree–decomposition for X such that the set of boundary vertices is indexed by the root of the tree[5]. By the term j^{th} *generation* we mean the set of graphs of $\mathcal{U}^t_{tree}$ of size j. This set is finite since a graph $G \in \mathcal{U}^t_{tree}$ of size j has a tree decomposition (T, B) such that $|V(T)| = j$ and $tw(G) \leq t$, so G can have at most $j(t+1)$ nodes and there are finitely many graphs with at most $j(t+1)$ nodes.

To clarify (b) we firstly extend the minor ordering $\leq_m$ to a minor ordering for t–boundaried graphs, $\leq_{\partial m}$. For $H, G \in \mathcal{U}^t_{tree}$, $H \leq_{\partial m} G$ if a graph isomorphic[6] to H can be obtained from G by a sequence of vertex deletions of non boundary vertices, edge deletions, and edge contractions of edges that have at least one non–boundary endpoint. In fact, $\leq_{\partial m}$ is a restriction of $\leq_m$, where we can apply the graph operations that procude a minor in a way that leaves the boundary vertices unaffected: if $H \leq_{\partial m} G$ then $B = B_H$, where $H = (V_H, E_H, B_H, f_H)$ and $G = (V, E, B, f)$. The $\leq_{\partial m}$ ordering can easily be shown to be a WQO on $\mathcal{U}^t_{large}$ by using the Graph Minor Theorem for edge–colored graphs. Let $X, Y \in \mathcal{U}^t_{tree}$, we define $X \leq Y$ if and only if $X \leq_{\partial m} Y$ and $X \sim Y$. Since there are finitely many equivalence classes of $\sim$ (by hypothesis) and $\leq_{\partial m}$ is a WQO, $\leq$ is a WQO on $\mathcal{U}^t_{tree}$.

And for (c), we say that there is *nothing new at time* j if none of the t–boundaried graphs of the j^{th} generation are minimal with respect to the search ordering $\leq$. A *stop signal* is detected at time $2j$ if there is nothing new at time i for $i = j, \ldots, 2j-1$ (see Figure 2.2).

To end the proof we need to prove the correctness of the algorithm. For this we need to prove that (1) a stop signal will be met eventually for every $t = 0, \ldots, B+1$, (2) the stop signal is valid, which means that all the obstructions in $\mathcal{U}^t_{tree}$ will be found until the stop signal will be met, and (3) all obstructions will be found by the end of Algorithm 2.

For (1), we know that $\leq$ is a WQO on $\mathcal{U}^t_{tree}$, so there is a finite number of minimal elements for the algorithm to search for.

To prove (2), which is the most crucial part of this proof, we have to show that if for a given t a stop signal is detected, then no obstruction for $\mathcal{F}$, has size greater than $2j$.

[5]In this proof all the trees of the tree–decompositions are as described in the definition of $size(X)$. That is, the root of the tree contains all the boundary vertices.

[6]If $G = (V, E, B, f)$ and $G' = (V', E', B', f')$ are two t–boundaried graphs, then G and G' are isomorphic if there is a bijection $g : V \to V'$, such that $\{v_1, v_2\} \in E \iff \{g(v_1), g(v_2)\} \in E'$, $v \in B \iff g(v) \in B'$ and for every $v \in B$, $f(v) = f'(g(v))$.

generation\tw	0	1	...	t	...	$B+1$
1				✓		
2				⋮		
⋮				✓		
j				nothing new		
⋮				⋮		
$2j-1$				nothing new		

Figure 2.2: Algorithm 2 searches the obstructions of $\mathcal{U}^t_{tree}$ by examining every generation for $\leq$–minimal graphs and stops its search when a *stop signal at time* $2j$ is detected. Thus, the search space is bounded and so the running time of Algorithm 2, which makes the problem of finding the obstruction set computable, provided that we have the information needed in Theorem 2.8.

We say that a graph X is a *prefix* of a graph G if the tree T_X of the tree–decomposition of minimum size for X is a rooted subtree of the tree T_G corresponding to the tree–decomposition of minimum size for G. Notice that we can substitute a prefix of a graph for another. In terms of tree decompositions, the substitution is done by replacing the rooted subtree corresponding to the old prefix with the rooted tree corresponding to the new prefix. This replacement is mirrored on the original graph, since the fact that the prefix graphs correspond to rooted trees, allow us to have the same result if we substitute the subgraph that corresponds to the old prefix for the graph which is the new prefix.

Let (T, B) be a tree–decomposition of minimum size for a counterexample H of our claim for proving (2). Since $size(H) > 2j$, there exists a prefix X of H such that $j \leq size(X) \leq 2j-1$. And since H is $\leq$–minimal, then X (and any prefix of H) must be minimal. This holds because if X is not minimal then there is an X' such that $X' < X$, that is $X' \leq X$, $X' \neq X$ and $X' \sim X$. We know that $\sim$ is a congruence, so $H' \sim H$, where H' is obtained by H by substituting the rooted subtree T_X of T with the rooted tree $T_{X'}$, where $(T_{X'}, B_{X'})$ and (T_X, B_X) are tree–decompositions, of minimum size, for X' and X respectively. By $X' < X$, we get that $H' < H$, which is a contradiction to our initial hypothesis that H is minimal. Since X is not minimal because a stop signal is detected at time $2j$, H is also not minimal which is a contradiction, because all obstructions are $\leq$–minimal graphs.

Finally, in order to show that by the end of algorithm 2 every obstruction will be found (3), we just need to prove that if $X \in \mathcal{O}$ then $\exists t \leq B+1$ such that $X \in \mathcal{M}_t$. Since $\mathbf{tw}(X) \leq B$, then $X \in \mathcal{U}^t_{tree}$ for some $t \leq B+1$. Also, $\sim$ refines the canonical $\mathcal{F}$–congruence and any proper minor of X belongs to $\mathcal{F}$. Thus X is $\leq$–minimal and the proof is complete. □

A pathwidth version of the theorem holds in essentially the same way. Instead of rooted trees we have paths with a distinct end, and X is a prefix of G if P_X has less nodes than P_G, where (P_X, B_X) and (P_G, B_G) are path decompositions of X and G

respectively. In addition, an immersion version of the theorem can be proven, again in the same way, but we have to define the ordering $\leq$ on t–boundaried graphs in terms of immersions. That is, $H \leq G$, iff $H \leq_{\partial im}$ G and $H \sim G$. And $\leq_{\partial im}$ is a restriction of $\leq_{im}$ that preserves the boundary. That is, $H \leq_{\partial im} G$ if a graph isomporphic to H can be obtained from G by applying a sequence of vertex deletions of non–boundary vertices, edge deletions and edge liftings. Finally, Algorithm 2 has been implemented and some results can be found in [9] and [8].

Knowing a bound on the treewidth of the obstructions was essential information for Algorithm 2, in terms of bounding the search space for the obstructions. However, neither such a bound is always known, nor a generic way to find it. So, in the next section we will try to overcome this difficulty by employing various *stop signals*.

2.3 Overcoming treewidth bound

Our first attempt to design an algorithm for obstruction set computation without the knowledge of a bound on the obstructions treewidth is, in a way, a generalization of Algorithm 2. The search is done in the same way but the whole prosess stops differently. For this we have to define a notion that will be usefull in the sequel, the *alarm*.

Definition 2.9. *An alarm for a minor–closed class $\mathcal{F}$ is a pair of computable functions:*

(1) $f_a : \mathbb{N} \to \mathbb{N}$, *and*

(2) $a : \mathbb{N} \to \{0, 1\}$, *satisfying:*

- *(a) (reliability) a(t)=1 if there is an obstruction $H \in \mathcal{O} \setminus \mathcal{O}^t$ of pathwidth more than $f_a(t)$*
- *(b) (eventual quiescence) $\exists t_0$ such that $\forall t \geq t_0$, $a(t) = 0$.*

We also have to point out that in this and the following sections of this chapter we will be working in $\mathcal{U}^t_{path}$.

Theorem 2.10. *If the following are known for a minor–closed graph class $\mathcal{F}$:*

(1) A decision algorithm for $\mathcal{F}$–membership.

(2) A decision algorithm for a finite–index congruence for $\mathcal{F}$ (the congruence can be either large or small).

(3) Algorithms for computing a and f_a for an alarm for $\mathcal{F}$.

then the obstruction set $\mathcal{O}$ of $\mathcal{F}$ can be computed.

Proof. For every t we compute $\mathcal{O}^t$ using the methods of the proof of Theorem 2.8 (adapted to pathwidth computations), namely (1) and (2). That is, we have two (while) loops, the outer to specify the pathwidth and the inner to specify the generation and at the end of every itteration of the outer loop, the underlying graphs of the $\leq$–minimal

elements of $\mathcal{U}^t_{path}$ enumerated that do not belong to $\mathcal{F}$ are stored in $\mathcal{O}^t$. For a certain pathwidth t, the inner loop stops exactly as described in the proof of Theorem 2.8. Since we don't know a bound on the pathwidth of the $\mathcal{F}$–obstructions we will employ a certain stop signal to halt the outer loop (the one that specifies the pathwidth).

We say that a t-stop signal is detected if

(a) $\mathcal{O}^i = \mathcal{O}^t$ for $i = t, \ldots, f_a(t)$, and

(b) a(t)=0

Using the t-stop signal we compute the $\mathcal{F}$–obstructions as described in Algorithm 3.

Algorithm 3 obs($\mathcal{F}$) computation

```
t ← 0
while a stop signal is not detected do
   compute O^i, for i = t, ..., f_a(t) and a(t)
   t ← t + 1
end while
return O^t
```

In order to prove that the algorithm is correct we have to show that (a) if $\mathcal{O} \neq \mathcal{O}^t$ then there will be no t-stop signal, and (b) eventually a t-stop signal will be met.
For (a) we have to examine two cases:
(1) There exists an obstruction $H \in \mathcal{O} \setminus \mathcal{O}^t$ for which $\mathbf{pw}(H) \leq f_a(t)$. In this case the first condition of the t-stop signal will fail.
(2) There exists an obstruction $H \in \mathcal{O} \setminus \mathcal{O}^t$ for which $\mathbf{pw}(H) > f_a(t)$. In this case, by the reliability of the alarm (definition 2.9) we have that $a(t) = 1$, so the second condition of the t-stop signal will fail.

For (b), let t' be the maximum pathwidth of the obstructions for $\mathcal{F}$. Then for $i = t', \ldots, f_a(t')$ we have that $\mathcal{O}^i = \mathcal{O}^{t'}$ (since for every $i \geq t'$, $\mathcal{O}^i = \mathcal{O}^{t'}$) and $a(t') = 0$, since there is no obstruction with pathwidth more than t'. Thus, a t'-stop signal will be met and Algorithm 3 will stop having correctly computed the obstruction set $\mathcal{O} = \mathcal{O}^{t'}$ of $\mathcal{F}$. □

Theorem 2.10 actually gives the general framework that we use for obstruction set computation. In that sense we can see theorem 2.8 as a special case of theorem 2.10, but our goal is to have an alarm that does not depend on the bound of the treewidth of the obstructions. To accomplish that we will employ a *terminating second–order congruence* as an alarm. Before that, we give the necessary definitions.

Definition 2.11. *The canonical second–order congruence for a minor–closed graph class $\mathcal{F}$, $\approx_{\mathcal{F}}$, is defined on finite subsets of t–boundaried graphs in $\mathcal{U}^t_{large}$ as follows: if $S_1, S_2 \subseteq \mathcal{U}^t_{large}$ then $S_1 \approx_{\mathcal{F}} S_2$ if and only if*

$$\forall Z \in \mathcal{U}^t_{large} : (\forall X_1 \in S_1 : X_1 \oplus Z \in \mathcal{F}) \iff (\forall X_2 \in S_2 : X_2 \oplus Z \in \mathcal{F})$$

Definition 2.12. *Let $\sim$ be an equivalence relation defined on finite subsets of $\mathcal{U}^t_{large}$. $\sim$ is a second–order congruence if $\approx_{\mathcal{F}}$ refines $\sim$, i.e. $\sim$ implies $\approx_{\mathcal{F}}$.*

Notice that a second–order congruence $\sim$ defines a first–order congruence when restricted to singletons: $\{A_1\} \sim \{A_2\}$, for $A_1, A_2 \in \mathcal{U}^t_{large}$. Thus the canonical second–order congruence restricted on signletons is actually the canonical first–order congruence (large canonical congruence).

Definition 2.13. *Let $\sim$ be a second–order congruence defined on a minor–closed graph class $\mathcal{F}$ and $S(A)$ denote all the t–boundaried graphs H, such that $H <_{\partial m} G$, for $A \in \mathcal{U}^t_{large}$. Then $\sim$ is called terminating if it satisfies the condition: $\exists t_0$ such that $\forall t \geq t_0$, if $A \in \mathcal{U}^t_{path}$ such that (1) $\mathbf{pw}(A) \geq t_0$ and (2) $|int(A)| \geq t_0$, then $\{A\} \sim \{S(A)\}$.*

On the above definition, $int(A)$ denotes the set of non–boundary vertices. In order to prove the next algorithm we need a structural lemma that first appeared in [7] as the *Wide Factor Lemma*, and was slightly adapted to our needs and proved in [6] as the *Fat Factor Lemma*.

Let B_h denote the complete binary tree of height h. B_h is a rooted tree, every vertex that is not a leaf has two children, every leaf has distance h from the root and B_h has $2^{h+1} - 1$ vertices. We also define $h(t) = \min\{h \in \mathbb{N} \mid \mathbf{pw}(B_h) \geq t\}$ and $f(t) = |V(B_{h(t)})|$. It can be shown that $f(t) = O(2^{2t})$. Finally, $f^{-1}(y) = \max\{x \in \mathbb{N} \mid f(x) \leq y\}$.

Lemma 2.14 (Wide Factor Lemma)**.** *Let H be and arbitrary undirected graph and t a positive integer. Then one of the two following statements must hold:*

(a) $\mathbf{pw}(H) \leq f(t) - 1$

(b) H can be factored in $H = A \oplus B$, where $A, B \in \mathcal{U}^{f(a)}_{large}$ and $\mathbf{pw}(A) > t$.

Additionally, if $f(t+1) > t' > f(t)$, then on of the following must hold:

(c) $\mathbf{pw}(A) \leq t' - 1$

(d) $H = A \oplus B$, $A \in \mathcal{U}^{t'}_{path}$, $B \in \mathcal{U}^{t'}_{large}$ and $\mathbf{pw}(A) > t$.

Lemma 2.15 (Fat Factor Lemma)**.** *There is a (known) recursive function $f(t) = O(2^{2t})$ such that if H is an arbitrary undirected graph then one of the following three statements must hold:*

(1) $\mathbf{pw}(A) \leq f(t) - 1$

(2) H can be factored in $H = A \oplus B$, where $A, B \in \mathcal{U}^{f(t)}_{large}$, $\mathbf{pw}(A) \geq t$, $|int(A)| \geq t$ and $A \in \mathcal{U}^{f(t)}_{path}$.

(3) H topologically contains K_t, the complete graph on t vertices.

Theorem 2.16. *If the following are known for a minor–closed graph class $\mathcal{F}$:*

(1) An algorithm for $\mathcal{F}$–membership

(2) A decision algorithm for a terminating second–order congruence $\approx$ for $\mathcal{F}$.

Then we can compute the obstruction set $\mathcal{O}$ for $\mathcal{F}$ (using the algorithms for (1) and (2) as subroutines).

Proof. Since we are trying to prove a 2.10–type theorem, $\mathcal{O}^t$ i.e. the obstructions of pathwidth at most t is computed in exactly the same way as descirbed in the proof of theorems 2.10 and 2.8. In particular, we compute all the minimal elements of $\mathcal{U}^t_{path}$ with respect to the order defined in the proof of theorem 2.8, i.e. $A \leq B$ iff $A \leq_{\partial m} B$ and $A \sim B$, where $\sim$ is the first–order congruence defined by the second–order congruence of (2) resticeted to singletons: $A \sim B$ iff $\{A\} \approx \{B\}$. This set is denoted by $\mathcal{M}^t$ and since $\leq$ is a WQO (by the GMT) and $\sim$ has finite index it is also finite. Then by using the algorithm for (1) we compute $\mathcal{O}^t$, which is the set of all underlying graphs of $\mathcal{M}^t$ that don't belong to $\mathcal{F}$.

To complete the proof we need to describe the alarm and prove that the resulting algorithm computes $\mathcal{O}$. In this direction, let $m(t)$ denote the maximum order of an obstruction H, such that $\mathbf{pw}(H) \leq t$, f be the function of the Fat Factor Lemma (lemma 2.15) and $t' > f(m(t))$. A *t–stop signal is witnessed at t'* if:
(a) $\mathcal{O}^i = \mathcal{O}^t$ for $i = t, \dots t'$, and
(b) $\forall A \in \mathcal{M}^{t'}$ such that $\mathbf{pw}(A) > f^{-1}(t') \geq m(t)$ and $|int(A)| > f^{-1}(t') \geq m(t) : \{A\} \approx S(A)$.

A *t–stop signal occurs* if there is a $t' > t$ as above at which a t–stop signal is witnessed. Using this stop signal we have a version of Algorithm 3 as follows.

Algorithm 4 obs($\mathcal{F}$) computation using the t–stop signal

```
t ← 0
while a t–stop signal is not detected do
  compute O^t and M^t
  t ← t + 1
end while
return O^t
```

To prove that Algorithm 4 computes $\mathcal{O}$, we have to show that if $\mathcal{O} \neq \mathcal{O}^t$ then no t–stop signal will occur. Suppose that a t–stop signal is witnessed at t' and let $H \in \mathcal{O} \setminus \mathcal{O}^t$. If $\mathbf{pw}(H) \leq t'$, then there can be no stop signal, because condition (a) of the t–stop signal will fail. And if $\mathbf{pw}(H) > t'$ and $t' > f(m(t))$ then by the Fat Factor Lemma (lemma 2.15) we have two cases:
(1) There exists a factorization $H = A \oplus B$ such that $A \in \mathcal{U}^{t'}_{path}$, $B \in \mathcal{U}^{t'}_{large}$, $\mathbf{pw}(A) > m(t)$ and $|int(A)| > m(t)$. Since A is an obstruction we have that $\forall A' < A$, $A \not\approx A'$, which implies that $A \in \mathcal{M}^{t'}$. Thus, $\{A\} \not\approx_{\mathcal{F}} S(A)$, which implies that $\{A\} \not\approx S(A)$, but this contradicts (b) of the t–stop signal.
(2) $K_{m(t)} \leq_t H$. In this case any obstruction in $\mathcal{O}^t$ is a topological minor of $K_{m(t)}$ which

is a topological minor of H, so any obstruction in $\mathcal{O}^t$ is a topological minor of H, which contradicts the minimality of H.

Thus, if $\mathcal{O} = \mathcal{O}^t$, which will eventually happen since $\leq$ is a WQO, a stop signal will be witnessed at time $t' = \max\{f(t_0), f(m(t))\}$, where t_0 is the termination constant, and the proof is complete. □

To accomplish our goal for designing an algorithm for obstruction set computation without the knowledge of a bound on the treewidth of the $\mathcal{F}$–obstructions we need to show that there exists a terminating second–order congruence.

Theorem 2.17 ([6])**.** *The canonical second–order congruence $\approx_{\mathcal{F}}$ for a minor–closed graph class $\mathcal{F}$ is terminating.*

2.4 Applications & final remarks

As an application of the two main results of [6] (theorems 2.8 and 2.16–by using $\approx_{\mathcal{F}}$ as the terminating second–order congruence) the problem of computing the obstruction set $\mathcal{O}$ for a union of two minor–closed graph classes $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ is studied. In this problem we are given the obstruction sets $\mathcal{O}_1$ and $\mathcal{O}_2$ of the minor–closed classes $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively and our objective is to compute $\mathcal{O}$. A weaker version of these problem is solved in [6], in which $\mathcal{O}$ can be computed by $\mathcal{O}_1$ and $\mathcal{O}_2$ and the use of Theorem 2.16, under the condition that $\mathcal{O}_1$ contains at least one tree[7].

In [1] the union problem, among others, is solved without the aforementioned condition. The authors present a different technique for obstruction set computation, in which they demand the knowledge of a formula in Monadic Second–Order Logic (MSO) defining a minor–closed graph class $\mathcal{F}$ and a bound on the treewidth of the obstructions of $\mathcal{F}$. We are going to present these methods and their applications (e.g. union problem) in the next chapter.

[7] Actually, the condition we need is that a tree is contained in at least one of $\mathcal{O}_1$ and $\mathcal{O}_2$.

Chapter 3

Trying Monadic Second–Order Logic

In this chapter we present some of the results in [1] by I. Adler, M. Grohe and S. Kreutzer, for obstruction set computation of minor–closed graph classes. The main theorem that is proven states that we can compute the obstruction set $\mathcal{O}$ of a minor–closed graph class $\mathcal{F}$ if we are given a formula $\varphi_{\mathcal{F}}$ in Monadic Second–Order Logic (MSO) that defines $\mathcal{F}$ and an upper bound on $width(\mathcal{F})$ (the bound on $width(\mathcal{F})$ is related to the bound on the treewidth of the obstructions for $\mathcal{F}$). By applying these methods, the authors solve (among others) the union problem. That is, given the obstruction sets $\mathcal{O}_1$ and $\mathcal{O}_2$ of two minor–closed graph classes $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively, we can effectively compute the obstruction set of their union $\mathcal{F}_1 \cup \mathcal{F}_2$.

3.1 Graphs and Logic

Lets recall the alternative definition of graph minors in section 1.1. We say that H is a minor of G, $H \leq_m G$, if there is a function that maps every vertex v of H to a connected subgraph $B_v \subseteq V(G)$, such that for every two distinct vertices v, w of H, B_v and B_w share no common vertex, and for every edge $\{u, v\}$ of H, there is an edge in G with one endpoint in B_v and one in B_u. The graph that is obtained by the union of all B_v such that $v \in V(H)$ and by the edges between B_v and B_u in G, if there exists an edge $\{v, u\}$ in H, is called a *model* of H in G. The model with the minimum number of vertices and edges is called *minimal model*. Obviously, if G' is a minimal model of H in G, then every B_v in G' is a tree with $deg_H(v)$ leaves.

Lemma 3.1. *Let $k > 0$ and $H \leq_m G$, with $|V(H)| = k$. Then any minimal model G' of H in G has $\mathbf{tw}(G') \leq k^2 + 1$.*

Proof. To prove this lemma we have to recall the definition of models. A model consists of $k = |V(H)|$ mutually disjoint connected subgraphs of G corresponding to the vertices of H and of $|E(H)|$ edges among them which correspond to the edges of H: there is an edge between B_u and B_v iff $\{u, v\}$ is an edge of H. In a minimal model, the B_v subgraphs are trees with at most $deg_H(v)$ leaves. Let G' be a minimal model of G. If we consider the graph that is obtained by G' by deleting all the edges among the B_v

trees, $v \in V(H)$, then the resulting graph is a forest (a union of trees) and has treewidth exactly 1. If we start putting back the edges that we removed, then every edge that we add can increase the treewidth at most by 1. Since we can add at most $\binom{k}{2}$ vertices ($k = |V(H)|$) and the fact that when we add all the edges that we removed the resulting graph will be again G', we have that $\mathbf{tw}(G') \leq \binom{k}{2} + 1 \leq k^2 + 1$. □

Corollary 3.2. *For all graphs H, G, if $H \leq_m G$ then there exists a subgraph G' of G with $H \leq_m G'$, satisfying $\mathbf{tw}(G') \leq |V(H)|^2 + 1$.*

By the Graph Minor Theorem and corollary 3.2, for every minor–closed graph class $\mathcal{F}$ there exists a w such that for every $G \notin \mathcal{F}$ there exists a subgraph G' of G, such that $\mathbf{tw}(G') \leq w$. To see why this holds, firstly consider that, by the GMT, the obstruction set $\mathcal{O}$ of a minor–closed graph class $\mathcal{F}$ is finite. And for every $G \notin \mathcal{F}$, there exists an $H \in \mathcal{O}$ such that $H \leq_m G$. Now if G' is the minimal model of H in G, by corollary 3.2 we have that $\mathbf{tw}(G') \leq |V(H)|^2 + 1$. Since $\mathcal{O}$ is finite, and if $w = \max\{|V(H)|^2 + 1 \mid H \in \mathcal{O}\}$ then for the (arbitrary) minor–closed class $\mathcal{F}$ and for every $G \notin \mathcal{F}$ there exists a subgraph G' of G such that $\mathbf{tw}(G') \leq |V(H)|^2 + 1 \leq w$, where H is an obstruction such that $H \leq_m G$. Finally, since $H \leq_m G'$, $G' \notin \mathcal{F}$.

Definition 3.3. *Let $\mathcal{F}$ be a minor–closed graph class. The least k such that for every $G \notin \mathcal{F}$ there exists a $G' \subseteq G$ with $G' \notin \mathcal{F}$ and $\mathbf{tw}(G') \leq k$ is called the width of $\mathcal{F}$ and is denoted by width($\mathcal{F}$).*

We now recall some definitions from Monadic Second Order Logic (MSO). An extended introduction to Logic can be found in [13, 24].

We call *signature* $\tau = \{R_1, \ldots, R_n\}$ any finite set of relation symbols R_i of any (finite) arity denoted by $\mathrm{ar}(R_i)$. For the language of graphs $\mathcal{G}$ we consider the signature $\tau_{\mathcal{G}} = \{V, E, I\}$ where V represents the set of vertices of a graph G, E the set of edges, and $I = \{(v, e) \mid v \in e \text{ and } e \in E(G)\}$ the incidence relation.

A τ-structure $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_n^{\mathfrak{A}})$ consists of a finite universe A, and the interpretation of the relation symbols R_i of τ in A, that is, for every i, $R_i^{\mathfrak{A}}$ is a subset of $A^{\mathrm{ar}(R_i)}$. For every graph we associate a relational structure $\mathfrak{G} := (V(G) \cup E(G), V^{\mathfrak{G}}, E^{\mathfrak{G}}, I^{\mathfrak{G}})$, over the signature $\tau_{\mathcal{G}}$, where $V^{\mathfrak{G}} = V(G)$, $E^{\mathfrak{G}} = E(G)$ and $I^{\mathfrak{G}} = \{(v, e) \mid e \in E(G) \text{ and } v \in e \cap V(G)\}$. A graph structure $\mathfrak{G}$ is a $\tau_{\mathcal{G}}$-structure, which represents the graph $G = (V, E)$. From now on, we abuse notation by treating G and $\mathfrak{G}$ equally.

In MSO formulas are defined inductively from atomic formulas, that is, expressions of the form $R_i(x_1, x_2, \ldots, x_{\mathrm{ar}(R_i)})$ or of the form $x = y$ where x_i, $i \in [\mathrm{ar}(R_i)]$, x and y are variables, by using the Boolean connectives $\neg, \wedge, \vee, \rightarrow$, and existential or universal quantification over individual variables and sets of variables. Furthermore, quantification takes place over vertex or edge variables or vertex-set or edge-set variables. Notice that in the language of graphs the atomic formulas are of the form $V(u), E(e)$ and $I(u, e)$, where u and e are vertex and edge variables respectively.

Definition 3.4. *A graph class $\mathcal{C}$ is* MSO-definable *if there exists an MSO formula $\varphi_{\mathcal{C}}$ in the language of graphs such that $G \in \mathcal{C}$ if and only if $G \models \varphi_{\mathcal{C}}$, that is, $\varphi_{\mathcal{C}}$ is true in the graph G (G models $\varphi_{\mathcal{C}}$).*

Lemma 3.5. *The class of graphs that contain a fixed graph H as a minor is MSO–definable by an MSO–formula φ_H.*

Proof. Let $H = (V, E)$. Then the MSO–sentence φ_H is defined as $\exists X_1 \dots \exists X_k \psi$ where ψ is the following sentence:

$$\left(\bigwedge_{i \neq j} X_i \cap X_j = \emptyset \wedge \bigwedge_i (\emptyset \subsetneq X_i \subseteq V \wedge \text{“}X_i \text{ is connected”} \wedge \right.$$

$$\left. \bigwedge_{\{v_i, v_j\} \in E(H)} \exists x_i \in X_i \exists x_j \in X_j \exists z((x_i, z) \in I \wedge (x_j, z) \in I) \right)$$

where “X_i is connected” can be expressed in MSO by the following formula:

$$\forall u \in X_i \forall v \in X_i \exists \{x_1, \dots, x_\ell\} \subseteq X_i \left(x_1 = u \wedge \bigwedge_{i=1}^{\ell - 1} \{x_i, x_{i+1}\} \wedge x_u = v \right)$$

□

Thus, $H \not\leq_m G$ iff $G \models \neg\varphi_H$. So if $\mathcal{O}$ is the obstruction set of a minor–closed class $\mathcal{C}$, then if $\varphi_{\mathcal{O}} \equiv \bigvee_{H \in \mathcal{O}} \varphi_H$ we have that $G \in \mathcal{C} \iff G \models \neg\varphi_{\mathcal{O}}$ and $\neg\varphi_{\mathcal{O}}$ defines $\mathcal{C}$.

The following theorem is basic for this chapter's approch on obstruction set computation algorithms.

Theorem 3.6 (Seese's Theorem [31])**.** *For every positive integer k, it is decidable given an MSO–formula whether it is satisfied by a graph G whose tree–width is upper bounded by k.*

Before we present the main theorem of this approach we have to extend the graph structures we defined earlier to structures that are defined by a graph and its tree–decomposition. Intuitively a tree–dec expansion is a pair of two structures, one defined by the graph G and one defined by a tree decomposition (T, B) of G.

Definition 3.7. *Let $G = (V, E)$ be a graph and $\mathcal{T} = (T, B)$ a tree decomposition of G. Let $\tau_{ex} = \{V, E, I, T, E_T, I_T, B\}$, where V, E, T, E_T are unary relation symbols and I, I_T, B are binary relation symbols.*
The tree–dec expansion TreeExp($G, \mathcal{T}$) of G and $\mathcal{T}$ is the τ_{ex}–structure

$$(V(G) \cup E(G) \cup V(T) \cup E(T), V^{\mathfrak{G}}, E^{\mathfrak{G}}, I^{\mathfrak{G}}, T^{\mathfrak{G}}, E_T^{\mathfrak{G}}, I_T^{\mathfrak{G}}, B^{\mathfrak{G}}),$$

where $V^{\mathfrak{G}}, E^{\mathfrak{G}}, I^{\mathfrak{G}}$ are defined as above and $T^{\mathfrak{G}} := V(T)$, $E^{\mathfrak{G}} := E(T)$, $I_T^{\mathfrak{G}} = \{(t, e) \mid e \in E(T) \text{ and } t \in e \cap V(T)\}$ and $B^{\mathfrak{G}} := \{(t, u) \mid t \in V(T) \text{ and } u \in B_t \cap V(G)\}$.

Lemma 3.8 ([1])**.** *(i) If G is a graph and $\mathcal{T} = (T, B)$ a tree decomposition of G of width k, then the treewidth of TreeExp(G, T) is at most $k + 2$.*

(ii) There is an MSO-sentence φ_{ex} so that for any τ_{ex}–structure $\mathfrak{G}$, $\mathfrak{G} \models \varphi_{ex}$ if, and only if, $\mathfrak{G}$ is a tree–dec expansion of a graph G.

The class of structures $\mathrm{TreeExp}(G, T)$ for which $G \in \mathcal{T}_k$ and $\mathcal{T}$ a tree–decomposition of G of width at most k, is denoted by $\mathrm{TreeExp}(T_k)$. It follows from Lemma 3.8 that $\mathrm{TreeExp}(T_k)$ is a class of structures of treewidth at most $k + 2$.

Corollary 3.9. *For all $k \in \mathbb{N}$, it is decidable if a given MSO–formula is satisfied in some $\mathfrak{G} \in TreeExp(T_k)$.*

3.2 Obstruction set computation

Definition 3.10. *A class of graphs $\mathcal{C}$ is layerwise–MSO definable, if for every $k \in \mathbb{N}$ we can compute an MSO–formula φ_k definining $C \cap \mathcal{T}_k$ in TreeExp($\mathcal{T}_k$).*

We are now ready to state and prove the main theorem of this approach for obstruction set computation.

Lemma 3.11 ([1])**.** *Let $\mathcal{C}$ be a minor–closed graph class. If $\mathcal{C}$ is layerwise MSO–definable and we are given an upper bound on its width $\mathcal{C}$, the obstruction set $\mathcal{O}$ of $\mathcal{C}$ is computable.*

Equivalnetly, there exists an algorithm that given $w \in \mathbb{N}$ and (an algorithm computing) a computable function $g : \mathbb{N} \to MSO$ so that there is a minor–closed class $\mathcal{C}$ with

- *for every $k \in \mathbb{N}$, $\varphi_k := g(k)$ defines $\mathcal{C} \cap \mathcal{T}_k$ in TreeExp($\mathcal{T}_k$) and*
- *the width of $\mathcal{C}$ is at most w,*

then the algorithm computes $\mathcal{O}$.

Proof. Firstly, we have to prove that for any given set $\mathcal{F} = \{F_1, \ldots, F_\ell\}$ we can decide if $\mathcal{F} = \mathcal{O}$ that is, if $\mathcal{F}$ is the obstruction set of $\mathcal{C}$. But $\mathcal{F} = \mathcal{O} \iff (G \in \mathcal{C} \iff \forall i \in [\ell] : F_i \not\leq_m G)$, or equivalently,

$$(G \in \mathcal{C} \implies \forall i \in [\ell] : F_i \not\leq_m G) \text{ and } (G \notin \mathcal{C} \implies \exists i \in [\ell] : F_i \leq_m G)$$
$$\iff$$
(i) $(G \in \mathcal{C} \wedge \exists i \in [\ell] : F_i \leq_m G)$ and (ii) $(G \notin \mathcal{F} \wedge \forall i \in [\ell] : F_i \not\leq_m G)$ are unsatisfiable.

To explain the above, the following hold:

$$\begin{aligned}
\varphi \leftrightarrow \psi &\iff (\varphi \to \psi) \wedge (\psi \to \varphi) \\
&\iff (\varphi \to \psi) \wedge (\neg\varphi \to \neg\psi) \\
&\iff (\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi) \\
&\iff \neg(\varphi \wedge \neg\psi) \wedge \neg(\neg\varphi \wedge \neg\psi) \\
&\iff (\varphi \wedge \neg\psi) \text{ and } (\neg\varphi \wedge \neg\psi) \text{ are unsatisfiable.}
\end{aligned}$$

Setting $\varphi \equiv (G \in \mathcal{C})$ and $\psi \equiv (\forall i \in [\ell] : F_i \not\leq_m G)$ we get the equivalences of the beginning of the proof. Thus, to decide if for a given set of graphs $\mathcal{F}$, $\mathcal{F} = \mathcal{O}$ we have to show that it is decidable that (i) and (ii) are unsatisfiable.

To prove that it is decidable that (i) is unsatisfiable, it suffices to check if $F_i \in \mathcal{C}$ for some $F_i \in \mathcal{F}$. If there is a $G \in \mathcal{C}$ such that $F_i \leq_m G$, for some $F_i \in \mathcal{F}$, then by lemma

3.1 there exists a $G' \subset G$ such that $\mathbf{tw}(G') \leq |V(F_i)|^2 + 1$ and $F_i \leq_m G'$. Since $\mathcal{C}$ is minor–closed, $G' \in C$. Thus, $(G \in \mathcal{C} \wedge \exists i \in [\ell] : F_i \leq_m G)$ satisfiable iff there exists G that satisfies the previous sentence such that $\mathbf{tw}(G) \leq k$, where $k := \max\{|V(F_i)|^2 + 1 : F_i \in \mathcal{F}\}$. By hypothesis, $\mathcal{C} \cap \mathcal{T}_k$ is MSO–definable in TreeExp($\mathcal{T}_k$) by $\varphi_{\mathcal{C}} := g(k)$. Let $\varphi_{\mathcal{F}} = \bigvee_{F_i \in \mathcal{F}} \varphi_{F_i}$, where φ_{F_i} is the formula defined in lemma 3.5. Every model G of $\varphi_{\mathcal{F}}$ has at least one graph $F \in \mathcal{F}$ as a minor. So, there exists $G \in \mathcal{C}$ such that $F_i \leq_m G$ for some $i \in [\ell]$. Equivalnetly, $\varphi_{\mathcal{C}} \wedge \varphi_{\mathcal{F}}$ is satisfiable in TreeExp($\mathcal{T}_k$), which is decidable by Seese's theorem (theorem 3.6).

To prove (ii), if there exists $G \notin \mathcal{C}$ such that $F_i \not\leq_m G$, $\forall i \in [\ell]$, since width($\mathcal{C}$)$\leq w$ there exists $G' \subseteq G$ such that $\mathbf{tw}(G') \leq w$ and $G' \notin \mathcal{C}$. Thus, $\forall i \in [\ell] : F_i \not\leq_m G'$. Let $\varphi_w := g(w)$ be the MSO–formula that defines $\mathcal{C} \cap \mathcal{T}_w$ in TreeExp($\mathcal{T}_w$), then there exists $G \notin \mathcal{C}$ such that $\mathbf{tw}(G) \leq w$ and $\forall i \in [\ell] : F_i \not\leq_m G$, iff, $\neg\varphi_w \wedge \neg\varphi\mathcal{F}$ is satisfiable in TreeExp($\mathcal{T}_w$), which is decidable in TreeExp($\mathcal{T}_w$) by Seese's theorem (theorem 3.6).

We define an ordering among sets of graphs. Let $\mathcal{F}_1 = \{F_1, \ldots, F_m\}$ and $\mathcal{F}_2 = \{H_1, \ldots, H_r\}$, then $\mathcal{F}_1 \leq_{\mathcal{C}} \mathcal{F}_2$ if either:

- $\sum_{i=1}^{m} |V(F_i)| < \sum_{i=1}^{r} |V(H_i)|$, or
- $\sum_{i=1}^{m} |V(F_i)| = \sum_{i=1}^{r} |V(H_i)|$ and $\sum_{i=1}^{m} |E(F_i)| < \sum_{i=1}^{r} |E(H_i)|$

Thus, by enumerating all finite sets of graphs $\mathcal{F}$ with respect to the ordering $\leq_{\mathcal{C}}$, the first set for which (i) and (ii) are unsatisfiable is the obstruction set of $\mathcal{C}$: $\mathcal{F} = \mathcal{O}$. The latter is also a description of an algorithm for computing the obstruction set $\mathcal{O}$ of $\mathcal{C}$:

Algorithm 5 obs($\mathcal{C}$) computation

$\mathcal{F} = \emptyset$
while (i) or (ii) is satisfiable for $\mathcal{F}$ **do**
  generate a $\leq_{\mathcal{C}}$–minimal set of graphs $\mathcal{F}'$ such that $\mathcal{F} \leq_{\mathcal{C}} \mathcal{F}'$ and $\mathcal{F}'$ has not been generated again
  $\mathcal{F} \leftarrow \mathcal{F}'$
end while
return $\mathcal{F}$

We are sure that the above algorithm terminates by the GMT and by this proof we are sure that it outputs the obstruction set of $\mathcal{C}$. □

3.3 Applications

Using lemma 3.11 it is shown that we can compute the obstruction sets of several classes such as $\mathcal{T}_k$, the class of graphs of treewidth at most k, $\mathcal{B}_k$, the class of graph of branch-width at most k [4] and $\mathcal{G}_k$, the class of graphs of genus at most k, by theorems 4.1, 4.2 and 4.3 respectively in [1]. Another theorem and the main combinatorial result of this

paper is given in theorem 5.1 of the paper, which states that if we are given the obstruction set of a minor–closed class $\mathcal{C}$, then we can compute the obstruction set of $\mathcal{C}^{apex}$, where[1] $G \in C^{apex} \iff \exists v \in V(G) : G \backslash v \in \mathcal{C}$. For proving the latter, Robertson and Seymour's *Trinity Lemma* is essential, as well as the irrelevant vertex technique [26, 29] play a key role on proving that $\mathcal{C}^{apex}$ has bounded width and since it is MSO–definable by $\varphi_{\mathcal{C}^{apex}}$. $\varphi_{\mathcal{C}^{apex}}$ is the following formula: $\exists v \in V \exists V' \exists E' (V' = V \setminus \{v\} \wedge E' = \{e \in E \mid v \in e \cap V\} \wedge \neg \left(\bigvee_{H_i \in \mathcal{O}_{\mathcal{C}}} \varphi'_{H_i} \right))$, where $\mathcal{O}_{\mathcal{C}}$ is the obstruction set of $\mathcal{C}$, $\varphi'_{H_i} \equiv \exists X_1 \ldots \exists X_k \psi$, $\psi \equiv [\bigwedge_{i \neq j} X_i \cap X_j = \emptyset \wedge \bigwedge_i (\emptyset \subseteq X_i \subseteq V' \wedge X_i \text{ is connencted }) \wedge \bigwedge_{\{v_i, v_j\} \in E(H_i)} \exists x_i \in X_i \exists x_j \in X_j \exists z((x_i, z) \in I(V', E') \wedge (x_j, z) \in I(V', E'))]$, $I(V', E') = \{(v, e) : e \in E' \wedge v \in e \cap V'\}$ and v_i are the vertices of H_i.

By applying the methods that where developed to prove that $\mathcal{C}^{apex}$ has bounded width, the authors prove that if we are given the obstruction sets $\mathcal{O}_1$ and $\mathcal{O}_2$ of two minor–closed graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, then $\mathcal{C}_1 \cup \mathcal{C}_2$ has bounded width and since $\mathcal{C}_1 \cup \mathcal{C}_2$ is MSO–definable by the formula $\neg \left(\bigvee_{H \in \mathcal{O}_1} \varphi_H \right) \vee \neg \left(\bigvee_{G \in \mathcal{O}_2} \varphi_G \right)$ we can apply lemma 3.11 and prove that $\mathcal{O}_{\mathcal{C}_1 \cup \mathcal{C}_2}$ is computable. This solves the union problem, and answers an open question from [6].

As a final remark, it is easy to see that for applying lemma 3.11, usually the easy task is to prove MSO–definability. The hard task is to prove that the class in study has bounded width, which we can easily understand by the use of the complex combinatorial results that were employed to show that $\mathcal{C}^{apex}$ has bounded width and to apply this methods to show that $\mathcal{C}_1 \cup \mathcal{C}_2$ has also bounded width (given $\mathcal{O}$ and, $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively). However, lemma 3.11 can be adapted to other well–quasi orderings. In the next chapter we will show that the methods of [1] that were presented in this chapter can be adapted to immersion–closed graph classes and by combining them with some other combinatorial results we can also solve the union problem for immersion–closed graph classes.

[1] We remind for this definition that $G \backslash v = (V \setminus \{v\}, E')$, where $E' = \{e \in E \mid v \notin e \cap V\}$.

Chapter 4

Immersion obstructions

In [1], I. Adler, M. Grohe and S. Kreutzer provide tools that allow us to use Seese's theorem (theorem 3.6), when an upper bound on the tree-width of the obstructions is known and an MSO–description of the graph class can be computed, in order to compute the obstruction sets of minor-closed graph classes. In this chapter we show how we can adapt their machinery to the immersion relation and prove that the treewidth of the obstructions of immersion–closed graph classes is upper bounded by some function that only depends on the graph class. This provides a generic technique to construct immersion obstruction sets when the explicit value of the function is known. Then, by obtaining such an explicit upper bound on the tree-width of the graphs in $\mathbf{obs}_{\leq_{im}}(\mathcal{C})$, where $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ and $\mathcal{C}_1$, $\mathcal{C}_2$ are immersion-closed graph classes whose obstruction sets are given, we show that the set $\mathbf{obs}_{\leq_{im}}(\mathcal{C})$ can be effectively computed. These results appeared in [17].

4.1 Preliminaries

We firstly recall and give some basic definitions. We say that H is an *immersion* of G (or H is *immersed* in G), $H \leq_{im} G$, if H can be obtained from a subgraph of G after a (possibly empty) sequence of edge lifts, where the *lift* of two edges $e_1 = \{x, y\}$ and $e_2 = \{x, z\}$ to an edge e is the operation of removing e_1 and e_2 from G and then adding the edge $e = \{y, z\}$ in the resulting graph. Equivalently, we say that H is an immersion of G if there is an injective mapping $f : V(H) \to V(G)$ such that, for every edge $\{u, v\}$ of H, there is a path from $f(u)$ to $f(v)$ in G and for any two distinct edges of H the corresponding paths in G are *edge-disjoint*, that is, they do not share common edges.

Additionally, if these paths are internally disjoint from $f(V(H))$, then we say that H is *strongly immersed* in G. As above, the function f is called a *model of H in G* and a model with minimal number of vertices and edges is called *minimal model*. A graph class $\mathcal{C}$ is called *immersion–closed*, if for every $G \in \mathcal{C}$ and every H with $H \leq_{im} G$ it holds that $H \in \mathcal{C}$. For example, the class of graphs $\mathcal{E}_t$ that admit a proper edge–coloring of at most t colors such that for every two edges of the same color every path between them contains an edge of greater color is immersion–closed. (See [3]). Two paths are

called *vertex-disjoint* if they do not share common vertices.

We will employ the ordering defined in the proof of lemma 3.11. The ordering $\leq$ is defined between finite sets of graphs as follows: $\mathcal{F}_1 \leq \mathcal{F}_2$ if and only if,

1. $\sum_{G\in\mathcal{F}_1} |V(G)| < \sum_{H\in\mathcal{F}_2} |V(H)|$ or
2. $\sum_{G\in\mathcal{F}_1} |V(G)| = \sum_{H\in\mathcal{F}_2} |V(H)|$ and $\sum_{G\in\mathcal{F}_1} |E(G)| < \sum_{H\in\mathcal{F}_2} |E(H)|$.

Definition 4.1. *Let $\mathcal{C}$ be an immersion-closed graph class. A set of graphs $F = \{H_1, \ldots, H_n\}$ is called (immersion) obstruction set of $\mathcal{C}$, and is denoted by $\mathbf{obs}_{\leq_{im}}(\mathcal{C})$, if and only if F is a $\leq$-minimal set of graphs for which the following holds: For every graph G, G does not belong to $\mathcal{C}$ if and only if there exists a graph $H \in F$ such that $H \leq_{im} G$.*

Recall (theorem 1.3) that by Robertson and Seymour's fundamental result in [30], the obstruction set of every immersion–closed graph class is finite.

Let r be a positive integer. An *r-approximate linkage* in a graph G is a family L of paths in G such that for every $r+1$ distinct paths $P_1, P_2, \ldots, P_{r+1}$ in L, it holds that $\bigcap_{i\in[r+1]} V(P_i) = \emptyset$ and each endpoint of the paths appears in exactly one path of L. We call these paths the *components* of the linkage. Let $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $(\beta_1, \beta_2, \ldots, \beta_k)$ be elements of $V(G)^k$. We say that an r-approximate linkage L, consisting of the paths $P_1, P_2, \ldots, P_k$, *links* $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $(\beta_1, \beta_2, \ldots, \beta_k)$ if P_i is a path with endpoints α_i and β_i, for every $i \in [k]$. The *order* of such linkage is k. We call an r-approximate linkage of order k, r-approximate k-linkage. Two r-approximate k-linkages L and L' are equivalent if they have the same order and for every component P of L there exists a component P' of L' with the same endpoints. An r-approximate linkage L of a graph G is called *unique* if for every equivalent linkage L' of L, $V(L) = V(L')$. When $r = 1$, such a family of paths is called *linkage*. Finally, a linkage L in a graph G is called *vital* if there is no other linkage in G joining the same pairs of vertices.

In [28], N. Robertson and P. Seymour proved a theorem which is known as The Vital Linkage Theorem. This theorem provides an upper bound for the tree-width of a graph G that contains a vital k-linkage L such that $V(L) = V(G)$, where the bound depends only on k. A stronger statement of the Vital Linkage Theorem was recently proved by K. Kawarabayashi and P. Wollan [20], where instead of asking for the linkage to be vital, it asks for it to be unique. Notice here that a vital linkage is also unique. As in some of our proofs (for example, the proof of Lemma 4.14) we deal with unique but not necessarily vital linkages we make use of the Vital Linkage Theorem in its latter form which is stated below.

Theorem 4.2 (The Unique Linkage Theorem[28, 20])**.** *There exists a computable function $w : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let L be a (1-approximate) k-linkage in G with $V(L) = V(G)$. If L is unique then $\mathbf{tw}(G) \leq w(k)$.*

Monadic second–order logic is going to play a key role in this approach too, so it is suggested to the reader to recall the definitions between definition 3.3 and lemma 3.8.

Lemma 4.3. *The class of graphs that contain a fixed graph H as an immersion is MSO–definable by an MSO–formula ϕ_H.*

Proof. Let $V(H) = \{v_1, v_2, \ldots, v_n\}$ and $E(H) = \{e_1, e_2, \ldots, e_m\}$. Let also ϕ_H be the following formula.

$$\phi_H := \exists E_1, E_2, \ldots, E_m \exists x_1, x_2, \ldots, x_n \Big[(\bigwedge_{i\in[n]} V(x_i)) \wedge (\bigwedge_{j\in[m]} E_i \subseteq E) \wedge (\bigwedge_{i\neq j} x_i \neq x_j) \wedge (\bigwedge_{p\neq q} E_p \cap E_q = \emptyset) \wedge (\bigwedge_{e_r=\{v_k,v_l\}\in E(H)} \mathrm{path}(x_k, x_l, E_r)) \Big],$$

where $\mathrm{path}(x, y, Z)$ is the MSO formula stating that the edges in Z form a path from x to y. This can be done by saying that the set Z of edges is connected and every vertex v incident to an edge in Z is either incident to exactly two edges of Z or to exactly one edge with further condition that $v = x$ or $v = z$. □

4.2 Computing immersion obstruction sets

In this Section we state the analogue of Lemma 2.2 in [1] (Lemma 4.4) and the analogue of Lemma 3.1 in [1] (Lemma 4.12, see also Lemma 3.11 in chapter 3) for the immersion relation. We include the necessary definitions and intermediate lemmata in order to give a sketch of the framework needed in order to prove the main result.

We first state the combinatorial Lemma of this Section.

Lemma 4.4. *There exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let H and G be graphs such that $H \leq_{im} G$. If G' is a minimal subgraph of G with $H \leq_{im} G'$ then $\mathbf{tw}(G') \leq f(|E(H)|)$.*

The proof of Lemma 4.4 is omitted as a stronger statement will be proved later on (Lemma 4.16). We continue by giving the necessary definitions in order to state the analog of Lemma 3.1 in [1] for the immersion relation.

Extension of MSO For convenience, we consider the extension of the signature $\tau_{\mathcal{G}}$ to a signature τ_{ex} that pairs the representation of a graph G with the representation of one of its tree–decompositions.

Definition 4.5. *If G is a graph and $\mathcal{T} = (T, B)$ is a tree-decomposition of G, τ_{ex} is the signature that consists of the relation symbols V, E, I of $\tau_{\mathcal{G}}$, and four more relation symbols V_T, E_T, I_T and B.*

A tree-dec expansion *of G and $\mathcal{T}$, is a τ_{ex}–structure*

$$\mathfrak{G}_{ex} = (V(G) \cup E(G) \cup V(T) \cup E(T), \\ V^{\mathfrak{G}_{ex}}, E^{\mathfrak{G}_{ex}}, I^{\mathfrak{G}_{ex}}, V_T^{\mathfrak{G}_{ex}}, E_T^{\mathfrak{G}_{ex}}, I_T^{\mathfrak{G}_{ex}}, B^{\mathfrak{G}_{ex}})$$

where $V_T^{\mathfrak{G}_{ex}} = V(T)$ represents the node set of T, $E_T^{\mathfrak{G}_{ex}} = E(T)$ the edge set of T, $I_T^{\mathfrak{G}_{ex}} = \{(v,e) \mid v \in e \cap V(T) \wedge e \in E(T)\}$ the incidence relation in T and $B^{\mathfrak{G}_{ex}} = \{(t,v) \mid t \in V(T) \wedge v \in B_t \cap V(G)\}$.

We denote by $\mathcal{C}_{\mathcal{T}_k}$ the class of tree–dec expansions consisting of a graph G with $\mathbf{tw}(G) \leq k$, and a tree decomposition (T,B) of G of width$(T,B) \leq k$.

Lemma 4.6 ([1])**.** *1. Let G be a graph and (T,B) a tree decomposition of it with width$(T,B) \leq k$. Then, the tree-width of the tree-dec expansion of G is at most $k+2$.*

2. There is an MSO-sentence $\phi_{\mathcal{C}_{\mathcal{T}_k}}$ such that for every τ_{ex}-structure $\mathfrak{G}$, $\mathfrak{G} \models \phi_{\mathcal{C}_{\mathcal{T}_k}}$ if and only if $\mathfrak{G} \in \mathcal{C}_{\mathcal{T}_k}$.

A classic result of Seese [31] (see Theorem 3.6) states that we can decide, for every $k \geq 0$, if an MSO-formula is satisfied in a graph G of $\mathbf{tw}(G) \leq k$. An immediate corollary of this result and Lemma 4.6 is the following.

Corollary 4.7. *We can decide, for every k, if an MSO-formula ϕ is satisfied in some $\mathfrak{G} \in \mathcal{C}_{\mathcal{T}_k}$.*

Theorem 4.8 ([1])**.** *For every $k \geq 0$, there is an MSO-sentence $\phi_{\mathcal{T}_k}$ such that for every tree-dec expansion $\mathfrak{G} \in \mathcal{C}_{\mathcal{T}_l}$ of G, for some $l \geq k$, it holds that $\mathfrak{G} \models \phi_{\mathcal{T}_k}$ if and only if $\mathbf{tw}(G) = k$.*

Definition 4.9. *A graph class $\mathcal{C}$ is* layer-wise MSO–definable*, if for every $k \in \mathbb{N}$ we can compute an MSO-formula ϕ_k such that $G \in \mathcal{C} \wedge \mathbf{tw}(G) \leq k$ if and only if $\mathfrak{G} \models \phi_k$, where $\mathfrak{G} \in \mathcal{C}_{\mathcal{T}_k}$ is the tree-dec expansion of G.*

Definition 4.10. *Let $\mathcal{C}$ be an immersion–closed graph class. The* width *of $\mathcal{C}$,* **width**$(\mathcal{C})$ *is the minimum positive integer k such that for every graph $G \notin \mathcal{C}$ there is a graph $G' \subseteq G$ with $G' \notin \mathcal{C}$ and $\mathbf{tw}(G') \leq k$.*

Note that Lemma 4.4 ensures that the width of an immersion-closed graph class is well-defined.

Observation 4.11. *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are immersion-closed graph classes then the following hold.*

1. For every graph $G \notin \mathcal{C}_1 \cup \mathcal{C}_2$, there exists a graph $G' \subseteq G$ such that $G' \notin \mathcal{C}_1 \cup \mathcal{C}_2$ and $\mathbf{tw}(G') \leq \max\{r(|E(H)|, |E(J)|) \mid H \in \mathbf{obs}_{\leq_{im}}(\mathcal{C}_1), J \in \mathbf{obs}_{\leq_{im}}(\mathcal{C}_2)\}$, where r is the function of Lemma 4.16 and thus,

2. *For every graph $G \notin \mathcal{C}_1$, there exists a graph $G' \subseteq G$ such that $G' \notin \mathcal{C}_1$ and $\mathbf{tw}(G') \leq \max\{f(|E(H)|) \mid H \in \mathbf{obs}_{\leq_{im}}(\mathcal{C}_1)\}$, where f is the function of Lemma 4.4.*

Finally, we state the analog of Lemma 3.1 in [1] for the immersion relation.

Lemma 4.12. *There exists an algorithm that, given an upper bound $l \geq 0$ on the width of a layer-wise MSO-definable class $\mathcal{C}$, and a computable function $f : \mathbb{N} \to MSO$ such that for every positive integer k, $f(k) = \phi_k$, where ϕ_k is the MSO-formula defining $\mathcal{C} \cap \mathcal{T}_k$, it computes $\mathbf{obs}_{\leq_{im}}(\mathcal{C})$.*

Remark 4.13. *We remark here that Lemma 4.12 provides a generic algorithm for computing the obstruction set of any immersion-closed graph class, given that the conditions stated are satisfied. However, notice that the above lemma implies that there is an algorithm that given an MSO formula ϕ and $k \in \mathbb{N}$, so that ϕ defines an immersion closed-graph class $\mathcal{C}$ of width at most k, computes the obstruction set of $\mathcal{C}$.*

The only missing ingredient towards our ultimate goal is an explicit upper bound on the tree-width of the obstructions of the graph class $\mathcal{C}_1 \cup \mathcal{C}_2$ where $\mathcal{C}_1$ and $\mathcal{C}_2$ are immersion-closed graph classes. In the following Section we prove that such a bound can be computed, given the obstruction sets of $\mathcal{C}_1$ and $\mathcal{C}_2$.

4.3 Treewidth Bounds for the Obstructions

We first prove the following generalization of the Unique Linkage Theorem.

Lemma 4.14 ([17])**.** *There exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let G be a graph that contains a 2-approximate k-linkage $\tilde{L}$ such that $V(\tilde{L}) = V(G)$. If $\tilde{L}$ is unique, then $\mathbf{tw}(G) \leq f(k)$.*

We remark that, the previous lemma holds for any graph G that contains an r-approximate k-linkage. This can be seen by substituting $(G \setminus T) \times K_2$ with $(G \setminus T) \times K_r$ in its proof.

We now state a lemma that provides the upper bound of a graph G, given the upper bound of its linear graph $L(G)$.

Lemma 4.15. *If G is a graph and k is a positive integer with $\mathbf{tw}(L(G)) \leq k$ then $\mathbf{tw}(G) \leq 2k + 1$.*

Before we proceed to the next lemma, we need to introduce the notion of an r-approximate k-edge-linkage in a graph. Similarly to the notion of an r-approximate linkage, an *r-approximate edge-linkage* in a graph G is a family of paths E in G such that for every $r + 1$ distinct paths $P_1, P_2, \ldots, P_{r+1}$ in E, it holds that $\cap_{i \in [r+1]} E(P_i) = \emptyset$. We call these paths the *components* of the edge-linkage. Let $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $(\beta_1, \beta_2, \ldots, \beta_k)$ be elements of $V(G)^k$. We say that an r-approximate edge-linkage E,

consisting of the paths $P_1, P_2, \ldots, P_k$, *links* $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $(\beta_1, \beta_2, \ldots, \beta_k)$ if P_i is a path with endpoints α_i and β_i, for every $i \in [k]$. The *order* of E is k. We call an r-approximate edge-linkage of order k, r-approximate k-edge-linkage. When $r = 1$, we call such a family of paths, an *edge-linkage*.

Lemma 4.16 ([17])**.** *There exists a computable function r such that the following holds. Let G_1, G_2 and G be graphs such that $G_i \leq_{im} G$, $i = 1, 2$. If G' is a minimal subgraph of G where $G_i \leq_{im} G'$, $i = 1, 2$, then $\mathbf{tw}(G') \leq r(|E(G_1)|, |E(G_2)|)$.*

Notice that Lemma 4.4 follows from Lemma 4.16 when we set G_2 to be the empty graph. Finally, we show that given two immersion closed graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$ the immersion-closed graph class $\mathcal{C}_1 \cup \mathcal{C}_2$ is layer-wise MSO-definable.

Observation 4.17. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be immersion-closed graph classes, then $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ is a layer-wise MSO-definable class defined, for every $k \geq 0$, by the formula*

$$\phi_k \equiv \left(\left(\bigwedge_{G \in \mathbf{obs}_{\leq_{im}}(\mathcal{C}_1)} \neg \phi_G \right) \vee \left(\bigwedge_{H \in \mathbf{obs}_{\leq_{im}}(\mathcal{C}_2)} \neg \phi_H \right) \right) \wedge \phi_{\mathcal{T}_k}$$

where ϕ_G and ϕ_H are the formulas described in Lemma 4.3, and $\phi_{\mathcal{T}_k}$ the formula of Theorem 4.8 .

We are now able to prove the Main Theorem.

Theorem 4.18. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two immersion-closed graph classes. If the sets $\mathbf{obs}_{\leq_{im}}(\mathcal{C}_1)$ and $\mathbf{obs}_{\leq_{im}}(\mathcal{C}_2)$ are given, then the set $\mathbf{obs}_{\leq_{im}}(\mathcal{C}_1 \cup \mathcal{C}_2)$ is computable.*

Proof. According to Observation 4.17, $\mathcal{C}_1 \cup \mathcal{C}_2$ is a layer-wise MSO-definable class, and according to Lemma 4.16 there is a bound on the width of $\mathcal{C}_1 \cup \mathcal{C}_2$. Therefore, Lemma 4.12 is applicable. □

4.4 Concluding remarks

In our study we delt with theorems for obstruction set compution for minor or immersion–closed graph classes. In both approaches, the algorithmic one and the one that employs MSO there is a common demand for a bound on the treewidth of the obstructions. However, in the algorithmic approach (Chapter 2), a way of overcoming this demand is described but when applied to the union problem we are not able to have a complete solution. On the other hand, by using the "MSO way" (Chapter 3) we can finally solve the union problem on its version for minor–closed graph classes. And by Chapter 4 we know that the theory of Chapter 3 can be adapted to the immersion relation and that we can solve the immersion version of the union problem.

We have to remark though, that these dependencies for obstruction set computation are crucial. For example knowing only the MSO–description of a minor (or immersion) closed graph class is not sufficient information for computing its obstruction set. On the

other hand, there is a need for a bound on the treewidth of the obstructions in both of the approaches studied and we do not know a generic procedure to compute one.

Finally, we are only concerned with matters of computability in our study, but all of our algorithms for obstruction set computation enumerate either graphs or sets of graphs, which is a rather "expensive" procedure in matters of time complexity. Since the aspect of computability is well studied, a direction for further research could involve ways to improve the time complexity of these algorithms or even new approaches that require less computational time. However, the obstruction sets are usually large sets. For example, a super–exponential lower bound on the size of the obstruction set for a certain family of graph classes is proven in [18] (see Theorem 1.7) which implies a super–exponential bound on the running time of every algorithm that computes it: only for "writing" an output of size n we need n amount of time.

Bibliography

[1] I. Adler, M. Grohe, S. Kreutzer, Computing Excluded Minors. S.-H.-Teng (Ed.), *SODA. SIAM.* pp. 641-650, 2008.

[2] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.* 23:1124, 1989.

[3] H. L. Bodlaender, J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, and Z. Tuza. Rankings of graphs. *SIAM J. Discrete Math.*, 11(1):168–181 (electronic), 1998.

[4] H.L. Bodlaender, D.M. Thilikos. Treewidth for graphs with small chordality. *Discrete Applied Mathematics*, Vol. 79, No. 1-3, pp. 45–61, 1997.

[5] H. L. Bodlaender. A partial k–arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209 (12): 145, 1998.

[6] K. Cattell, M. Dinneen, R. Downey, M. R. Fellows, M. Langston. On computing graph minor obstruction sets. *Theoretical Computer Science* 233, 107-127, 2000.

[7] K. Cattell, M. J. Dinneen and M. R. Fellows. A simple linear time algorithm for finding path decompositions of small width. *Information Processing Letters 57* , 197–203, 1996.

[8] K. Cattell, M. J. Dinneen and M. R. Fellows. Obstructions to within a few vertices or edges of acyclic.*Proceedings WADS '95*, Springer–Verlag, Lecture Notes in Computer Science vol. 995, 415–427, 1995.

[9] K. Cattell and M. J. Dinneen. A characterization of graphs with vertex cover up to five. *Proceedings ORDAL '94*, Springer–Verlag, Lecture Notes in Computer Science vol. 831, 86–99, 1994.

[10] B. Courcelle, R. G. Downey, and M. R. Fellows. A note on the computability of graph minor obstruction sets for monadic second order ideals. *Journal of Universal Computer Science*, 3:11941198, 1997.

[11] B. Courcelle and J. Lagergren. Equivalent definitions of recognizability for graphs of bounded tree-width. *Mathematical Structure in Computer Science*, to appear.

[12] R. Diestel. Graph Theory. *Graduate Texts in Mathematics*, Spirnger–Verlag, 4^{th} edition, 2010.

[13] H. B. Enderton. A Mathematical Introduction to Logic. *Academic Press*, New York, 1972.

[14] J. Erickson. Computational Topology. *Class notes, Department of Computer Science, University of Illinois, Urbana–Champaign.*
`http://compgeom.cs.uiuc.edu/~jeffe/teaching/comptop/`

[15] M. R. Fellows and M. A. Langston. An analogue of the Myhill–Nerode theorem and its use in computing finite–basis characterizations. *Proc. Symposium on the Foundations of Computer Science (FOCS)*, IEEE Press, 520–525, 1989.

[16] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial time algorithms. *Proc. Symposium on the Theory of Computing (STOC)*, ACM Press, 501–512, 1989.

[17] A. Giannopoulou, I. Salem, D. Zoros. Effective Computation of Immersion Obstructions for Unions of Graph Classes. 13^{th} *Scandinavian Symposium and Workshops on Algorithm Theory*, 2012.

[18] A. Giannopoulou, D. M. Thilikos. Obstructions for Tree–depth. *Electronic notes in Discrete Mathematics* 34, 249–253, 2009.

[19] J. Hopcroft and R.E. Tarjan. Efficient planarity testing. *Journal of the Association for Computing Machinery* 21 (4): 549–568, 1974.

[20] K. Kawarabayashi and P. Wollan. A shorter proof of the graph minor algorithm: the unique linkage theorem. In Leonard J. Schulman, editor, *STOC*, pages 687–694. ACM, 2010.

[21] W. Kocay. The Hopcroft–Tarjan Planarity Algorithm. *Unpublished.*
`http://www.combinatorialmath.ca/G&G/articles/planarity.pdf`

[22] J. B. Kruskal. Well–quasi–ordering, the Tree Theorem, and Vazsonyps conjecture. *Trans. Amer. Math. Soc.* 95, pp. 210225, 1960.

[23] L. Lovasz. Graph Minor Theory. *Bulletin (New Series) of the American Mathematical Society.* Volume 43, number 1, pages 75–86, 2005.

[24] E. Mendelson. Introduction to Mathematical Logic. 2nd ed. *Van Nostrand–Reinhold*, New York, 1978.

[25] N. Robertson and P. D. Seymour, Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms* 7(3):309322, 1986.

[26] N. Robertson and P. D. Seymour, Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65110, 1995.

[27] N. Robertson and P. D. Seymour, Graph minors XX. Wagners conjecture. *Journal of Combinatorial Theory, Series B*, 92:325357, 2004.

[28] N. Robertson and P. D. Seymour. Graph minors XXI. Graphs with unique linkages. *J. Comb. Theory, Ser. B*, 99(3):583–616, 2009.

[29] N. Robertson and P. D.Seymour. Graph minors XII. Irrelevant vertices in linkage problems. *submitted for publication*, 2006.
`http://www.math.princeton.edu/~pds/papers/GM22/GM22.pdf`

[30] N. Robertson and P. D. Seymour. Graph minors XXIII. Nash–Williams' immersion conjecture, *Journal of Combinatorial Theory, Series B* 100 (2): 181205, 2010.

[31] D. Seese. The structure of models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53:169195, 1991.