Model Checking for Successor-Invariant First-Order Logic on Minor-Closed Graph Classes

Kord Eickmeyer
National Inst. of Informatics, Tokyo
eickmeye@nii.ac.jp

Ken-ichi Kawarabayashi
National Inst. of Informatics, Tokyo,
and JST, ERATO,
Kawarabayashi Large Graph Project
k_keniti@nii.ac.jp

Stephan Kreutzer
Technical University Berlin
School of Elect. Eng. and Computer Science
stephan.kreutzer@tu-berlin.de

***Abstract*—Model checking problems for first- and monadic second-order logic on graphs have received considerable attention in the past, not the least due to their connections to problems in algorithmic graph structure theory. While the model checking problem for these logics on general graphs is computationally intractable, it becomes tractable on important classes of graphs such as those of bounded tree-width, planar graphs or more generally, classes of graphs excluding a fixed minor.**

It is well known that allowing an order relation or successor function can greatly increase the expressive power of the respective logics. This remains true even in cases where we require the formulas to be order- or successor-invariant, that is, while they can use an order relation, their truth in a given graph must not depend on the particular ordering or successor function chosen.

Naturally, the question arises whether this increase in expressive power comes at a cost in terms of tractability on specific classes of graphs. In LICS 2012, Engelmann et al. studied this problem and showed that order-invariant monadic second-order logic (MSO) remains tractable on the same classes of graphs than MSO without an ordering. That is, adding order-invariance to MSO essentially comes at no extra cost in terms of model checking complexity. For successor-invariant first-order logic something similar should be true. However, they only managed to show that successor-invariant first-order logic is tractable on the class of planar graphs which is very far from the best tractability results currently known for first-order logic.

In this paper we significantly improve the latter result and show that successor-invariant first-order logic is tractable on any class of graphs excluding a fixed minor. This is much closer to the best results known for FO without an ordering. The proof relies on the construction of k-walks in suitable supergraphs of the input graphs, i.e., walks which visit every vertex at least once and at most k times, for some k depending on the excluded minor H. The supergraphs may in general contain H minors, but they still exclude some possible larger minor H', so by results of Flum and Grohe [20] model checking on these graphs is still fixed-parameter tractable.

I. Introduction

Studying logics as foundation of query or specification languages has a long history in computer science. For instance, first-order logic is the logical foundation of the standard database query language SQL and monadic second-order logic plays a crucial rôle in formal language theory as, by Büchi's fundamental result [6], monadic second-order logic (MSO) in strings is equivalent to regular expressions and finite automata. MSO also plays an important rôle in algorithmic graph structure theory, as common algorithmic problems on graphs can very elegantly be formalised in MSO.

This work was supported by a fellowship of the first author within the FIT-Programme of the German Academic Exchange Service (DAAD).

Many interesting problems on graphs have been shown to be NP-complete. However, for problems such as finding minimal dominating sets or 3-colourings of graphs, efficient algorithms can be retained on restricted classes of graphs such as trees or planar graphs. A huge body of research has therefore been developed identifying graph classes on which important problems can be solved efficiently and developing the algorithmic and graph structural tools to establish such algorithms. In this context, the concept of *tree-width* has proved to be particularly important and successful. The tree-width of a graph associates with any graph a natural number, its tree-width $\mathrm{tw}(G)$, which measures how close a graph is to being a tree. Many hard algorithmic problems on graphs become tractable on graph classes of bounded tree-width, i.e., with a uniform bound on the tree-width of all members of the class, and a huge number of papers establish linear time algorithms for graph problems on graph classes of bounded tree-width (see e.g. [14], [3], [4] and references therein).

The concept of tractability in this context can best be stated in the framework of parameterised complexity theory (see [21], [14]). A *parameterised problem* P takes as input pairs (w, k) where w is a word over some alphabet, such as a graph or a database, and $k \in \mathbb{N}$. Here, k is called the *parameter*. P is called *fixed-parameter tractable* (FPT), if there is a constant $c \in \mathbb{N}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm solving P which on input (w, k) takes time $f(k) \cdot |w|^c$. Obviously, the concept of fixed-parameter tractability crucially relies on the choice of parameter. Usually, one either takes the solution size as parameter, or some graph structural parameter such as tree-width. For instance, it is known that 3-Colourability, the dominating set problem and many other problems are fixed-parameter tractable with respect to the tree-width as parameter. In the context of logical evaluation problems, i.e. the problem, given a structure such as a graph or database G and a formula φ, to decide whether $G \models \varphi$, we usually take the size of the formula φ as parameter. Hence, we aim for polynomial evaluation algorithms where only the constants but not the degree of the polynomial depend on the formula.

In 1992, Courcelle [7] proved his much celebrated result that every graph problem definable in monadic second-order logic can be decided in linear time on any class of graphs of bounded tree-width. In this context, one usually distinguishes between two variants of MSO, called MSO_1 and MSO_2. MSO_2 is the extension of first-order logic (FO) by quantification over sets of edges as well as sets of vertices. MSO_1, on the other hand, is the extension of FO by quantification over sets of vertices only. More precisely, Courcelle's theorem can be phrased as follows: there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that deciding whether an MSO_2-formula φ is true in a given graph G can be done in time $f(\mathrm{tw}(G) + |\varphi|) \cdot |G|$. As many graph problems can be formalised in MSO, this has proved to be an extremely useful and widely used strategy for obtaining linear time algorithms for many problems. See e.g. [26], [4] and references therein. See also [25], [27] where Courcelle's theorem is used in a highly non-trivial way that cannot be replaced easily by an algorithmic method. While Courcelle's theorem has mostly been used as a quick way of proving that linear time algorithms exist, more recently it has been shown to be practically applicable and very promising implementations have emerged. See [32] and the project webpage [36].

Courcelle's theorem has been extended in many ways. One direction is to consider extensions of MSO on classes of bounded tree-width which make the logic based approach to efficient algorithms applicable to optimisation and counting problems. See e.g. [1], [10].

As shown in [33], [35], Courcelle's theorem itself can essentially not be extended beyond classes of graphs of bounded tree-width. Hence, for more general classes of graphs one needs to consider less expressive logics than MSO_2. In [9], combined with [29], Courcelle et al. showed that a variant of MSO called MSO_1 is fixed-parameter tractable on any class of graphs of bounded clique-width. Seese [41] showed that first-order logic is fixed-parameter tractable on any class of graphs of bounded maximum degree. Frick et al. [22] extended this to show that FO is FPT on any class of graphs of bounded local tree-with. Furthermore, Flum et al. [19] showed the same result for classes of graphs excluding a fixed minor. More recently, both results have been unified and extended in [11] to classes of graphs locally excluding a minor. The most general result of this form known to date is tractability of first-order model checking on classes of graphs of locally bounded expansion [15]. As far as first-order model checking is concerned, it was shown in [34] that FO model checking is essentially not FPT on any class of graphs which is somewhere dense, i.e., not nowhere dense. Hence, in this line of research, the main open problem is tractability of FO model checking on classes of graphs which are nowhere dense.

As mentioned above, Courcelle's theorem has proved to be extremely useful in obtaining tractability results for problems on graph classes of bounded tree-width. Tree-width and many other graph structural properties are defined in terms of graph decompositions. For instance, tree-width is defined in terms of tree-decompositions, a recursive, tree-like, decomposition of a graph into subgraphs of constant size. This decomposition allows to apply dynamic programming techniques to solve algorithmic graph problems efficiently, similar to standard methods for solving these problems on trees. Many other structural parameters are similarly defined by means of suitable decompositions or embeddings into surfaces. For logic based methods in this area, it is often very useful if the corresponding decompositions can themselves be defined in the logic itself. As pointed out, e.g., in [8], this can often be done if the graph is equipped with an order relation. The problem is that any logic is invariant under automorphisms, i.e., cannot distinguish between automorphic vertices in a graph, whereas most decompositions will distinguish between them. If an ordering of the graph is given, then this problem disappears so that many decompositions become definable. This applies for instance to computing planar embeddings in MSO, chord diagrams of circle graphs, rank decompositions of ordered cographs, modular decompositions or split decompositions.

As all decompositions are defined on the graph itself, and not on its ordered version, it is irrelevant for the definability of the decompositions which order is used on the graph. As long as it is a linear order, the constructions work. This motivates the study of *order-invariant* MSO and FO, a concept that has previously been studied in database theory. In order-invariant MSO or FO we are allowed to use a linear order predicate $\leq$ in logical formulas, with the requirement that the truth of a formula in a structure $\mathfrak{A}$ must not depend on the choice of order. More formally, if $\mathfrak{A}$ is a logical structure over a signature σ s.t. $\leq \notin \sigma$, $\varphi \in \mathrm{MSO}[\sigma \cup \{\leq\}]$, and $\leq_1, \leq_2$ are linear orders on $\mathfrak{A}$ then $(\mathfrak{A}, \leq_1) \models \varphi$ if, and only if, $(\mathfrak{A}, \leq_2) \models \varphi$. A related concept to order-invariance is *successor-invariance*, where we are allowed to use a successor function but again the truth must not depend on the actual choice of the successor function. For MSO, order-invariance and successor-invariance are equivalent, but for FO the two concepts are different as the ordering is not definable from a successor function. As argued above, order-invariance is exactly what is needed to define various types of graph decompositions in logics, making this an interesting concept to study in logical approaches to graph algorithms.

Order-invariant logics where originally studied in a database context. It is easily seen that order-invariant MSO is more expressive than MSO, as for instance even cardinality becomes definable on any structure. Gurevich (in an unpublished note) was the first to observe that order-invariant FO is more expressive than plain FO. It is known that order-invariant FO collapses to FO on trees [2], [39], and that order-invariant FO is a subset of MSO on graphs of bounded degree and on graphs of bounded tree-width [2]. By a result of Rossman [40], we know that already the expressive power of successor-invariant FO is stronger than that of plain FO.

As mentioned above, much work has gone into studying the (parameterised) complexity for MSO and FO on specific classes of graphs. Given the usefulness of order-invariant logics in the context of graph algorithms, it is a natural question to study the complexity of the model checking problems for order- and successor-invariant logics on classes of graphs, i.e. to analyse in how that the increased expressive power of order- or successor-invariant logics effects the tractability of these logics on specific classes of graphs.

In [18], Engelmann et al. studied this problem and showed that order-invariant MSO_2 is FPT on classes of graphs of bounded tree-with and MSO_1 is fixed-parameter tractable on classes of graphs of bounded clique-width. Hence, we get exactly the same tractability results for order-invariant MSO than plain MSO. Moreover, they were able to reduce the problem for order-invariant MSO to the case for plain MSO so that the same model checking algorithms can be used. In particular, order-invariance could be added to implementations such as [36] with minimal overhead.

We conjecture that the same should be true for successor-invariant first-order logic, i.e. we conjecture that successor-invariant FO should be tractable on the same classes of graphs as plain FO. In [18] this was shown for planar graphs. In this paper we prove a huge step towards the conjecture by extending this result significantly to any class of graphs excluding a fixed minor. Note that here we apply the structural restriction, i.e., excluding a minor only to the graph and not to the graph plus its successor relation. Graph classes excluding a fixed minor are much more general than planar graphs and include many interesting examples of graphs. Our result here brings tractability of successor-invariant FO reasonably close to the best tractablity results known for plain FO. There is, however, still a gap between classes excluding a fixed minor and those of locally bounded expansion. We leave this for future research.

An important and very useful aspect of our technique is that we can again reduce the model checking problem for successor-invariant FO on graph classes excluding a fixed minor to the model checking problem for plain FO on classes excluding a fixed minor. This allows to reuse the algorithm established for plain FO and also means that successor-invariance could easily be added to any implementation of these results.

Our algorithm works as follows. Let $\mathcal{C}$ be a class of graphs excluding a fixed minor H. We show that there is a graph H' and a constant $k \geq 1$ such that for every graph $G \in \mathcal{C}$ we can construct in polynomial time a *k-walk* P in G (i.e., a walk P which visits every vertex at least once and at most k times) such that adding the edge set of P to G yields a coloured graph G' which excludes the graph H' as minor. Furthermore, there is an FO-interpretation that constructs in G' a copy of the graph G together with a successor-relation S. Now, given a graph $G \in \mathcal{C}$ and a successor-invariant first-order formula φ, we can verify whether $G \models \varphi$ as follows. We first construct in polynomial time the graph G' as before. Applying the interpretation yields a successor-relation S on G but also a formula $\varphi' \in \mathrm{FO}$ which does not use any successor relation such that $(G, S) \models \varphi$ if, and only if, $G' \models \varphi'$. As φ is successor-invariant, the truth of φ in G does not depend on the particular choice of the successor relation. Hence we can verify whether $G \models \varphi$ by checking whether $G' \models \varphi'$. As G' excludes the graph H' as a minor, this can be done using the known algorithms for graph classes excluding a fixed minor [19].

Organisation. We fix our notation and recall concepts from logic and graph theory needed in the sequel in Section II. In Section III we develop the main graph theoretical tools and show that for any graph H there is a constant $k \geq 1$ and a graph H' such that any graph G excluding H as a minor has a k-walk P such that $G \cup P$ excludes H'. In Section IV we establish the logical tools needed for our result, in particular the first-order interpretation defining a successor relation from the k-walk computed in Section III. Finally, in Section V we combine these to prove the main result of this paper.

II. Preliminaries and Notation

For a natural number n we let $[n]$ denote the interval $\{1, \ldots, n\}$.

A. Logics

We will be dealing with finite structures over finite, relational vocabularies. Thus a *vocabulary* σ is a finite set of relation symbols R, each with an associated *arity* $a(R)$, and a *σ-structure* A consists of a finite set $V(A)$ (the *universe*) and relations $R(A) \subseteq A^{a(R)}$ for all $R \in \sigma$. For vocabularies $\sigma \subseteq \tau$ and a σ-structure A, a *τ-expansion* B is a τ-structure with $V(A) = V(B)$ and $R(B) = R(A)$ for all $R \in \sigma$.

We use standard definitions for first-order logic (FO), cf. [17], [16], [37]. In particular, $\perp$ and $\top$ denote false and true, respectively. Let σ be a vocabulary and $\mathrm{succ} \notin \sigma$ a new binary relation symbol. We set $\sigma_{\mathrm{succ}} := \sigma \cup \{\mathrm{succ}\}$ and say that succ is interpreted by a *successor relation* in a σ_{succ}-structure B if $\mathrm{succ}(B)$ is the graph of a cyclic permutation on $V(B)$. An $\mathrm{FO}[\sigma_{\mathrm{succ}}]$-formula φ is called *successor-invariant* if for all σ-structures A and all σ_{succ}-expansions B, B' of A in which succ is interpreted by a successor relation we have

$$B \models \varphi \quad \Leftrightarrow \quad B' \models \varphi,$$

when all free variables of φ are interpreted identically in B and B'. In this case we say that $A \models \varphi$ if $B \models \varphi$ for one such expansion B (equivalently for all such expansions).

Note that another common definition of successor relation is to require $\mathrm{succ}(A)$ to be of the form

$$\{(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)\}$$

for some enumeration $V(A) := \{a_1, \ldots, a_n\}$ of the elements of $V(A)$. This differs from our definition in that we require $(a_n, a_1) \in \mathrm{succ}(A)$ as well, eliminating the somewhat artificial status of the first and last element. The definition of successor-invariant FO is not affected by this, though the quantifier rank of formulas may change.

B. Graphs

We will be dealing with finite simple (i.e., loop-free and without multiple edges) graphs, cf. [13], [43] for an in-depth introduction. Thus a *graph* $G = (V, E)$ consists of some finite set V of *vertices* and a set $E \subseteq \binom{V}{2}$ of *edges*. We write

$uv \in E$ for $\{u,v\} \in E$. For a set $U \subseteq V$ we denote the *induced subgraph* on U by $G[U]$, i.e., the graph (U, E') with

$$E' := \{uv \mid u, v \in U \text{ and } uv \in E\}.$$

For ease of notation we occasionally blur the distinction between a set U of vertices and the induced subgraph of this set. The union $G \cup H$ of two graphs $G = (V, E)$ and $H = (U, F)$ is defined as the graph $(U \cup V, E \cup F)$. For a set U of vertices, $K[U]$ denotes the complete graph (or *clique*) with vertex set U. For $k \in \mathbb{N}$, we denote the k-clique $K[[k]]$ by K_k.

For $k \geq 1$, a *k-walk* through a graph $G = (V, E)$ is a surjective mapping $w : [\ell] \to V$ for some $\ell \in \mathbb{N}$ such that

- $w(i)w(i+1) \in E$ for all $1 \leq i < \ell$,
- $|\{i \in [\ell] \mid w(i) = v\}| \leq k$ for all $v \in V$

The number ℓ is called the *length* of the walk.

A *surface* Σ is a compact Hausdorff space in which every point has a neighbourhood homeomorphic to the real plane $\mathbb{R}^2$. A *path* in Σ is a continuous mapping $\gamma : [0,1] \to \Sigma$ which is injective except for possibly $\gamma(0) = \gamma(1)$, in which case we call the path *closed*. A *drawing* Π of a graph $G = (V, E)$ on a surface Σ associates a point $\pi(v) \in \Sigma$ to every $v \in V$ and a path $\gamma := \pi(e)$ to every edge $e = uv$ such that $\gamma(0) = \pi(u)$ and $\gamma(1) = \pi(v)$ or the other way around and such that no two such paths share an interior point (i.e., a point $\gamma(x)$ for some $0 < x < 1$), and no interior point equals $\pi(v)$ for some $v \in V$. A graph G is called *embeddable* into some surface Σ iff such an embedding exists. A graph which is embeddable into the sphere is called *planar*.

A drawing Π is called *cellular* if every connected component of $\Sigma - \Pi$ is homeomorphic to an open disc. (Here, we identify Π with the subset $\{\pi(v) \mid v \in V\} \cup \bigcup_{e \in E} \operatorname{im}(\pi(e))$ of Π.) Such drawings can be described combinatorially by so-called *2-cell embedding schemes*, cf. [38, Ch. 3]. When we speak of a 2-cell embedding (G, Π) of a graph G in an algorithmic context, we mean a suitable representation of such an embedding scheme.

A *separation* of a graph $G = (V, E)$ is a pair (A, B) of nonempty subsets $A, B \subseteq V$ such that $V = A \cup B$ and there is no edge in G between any $a \in A \setminus B$ and $b \in B \setminus A$. For $c \in \mathbb{N}$, a graph is called c-connected if there is no separation (A, B) with $|A \cap B| < c$. For $c = 1$ we just say *connected*. We will need the following theorem about the existence of 2-walks in 3-connected planar graphs, cf. [23], [24]:

Theorem 2.1 (Gao and Richter)**.** *Let $G = (V, E)$ be a 3-connected planar graph. Then there is a 2-walk w through G, and it can be computed from G in polynomial time.*

Gao and Richter do not explicitly mention the polynomial time computability of the 2-walk, but there are polynomial time algorithms for computing planar embeddings (cf. [30], [5]), and given such an embedding the proof in [23] is easily seen to be constructive.

A *tree* is a connected acyclic graph, a *path* is a tree in which every vertex has degree at most two. A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, \mathcal{V})$ consisting of a tree $\mathcal{T} = (T, F)$ and a mapping $\mathcal{V} : T \to 2^V, t \mapsto \mathcal{V}_t$ such that

- $\bigcup_{t \in T} \mathcal{V}_t = V$,
- for every edge $uv \in E$ there is a $t \in T$ with $u, v \in \mathcal{V}_t$, and
- for every $v \in V$ the set $\{t \in T \mid v \in \mathcal{V}_t\}$ is a subtree of $\mathcal{T}$ (i.e., connected).

The sets $\mathcal{V}_t$ are called the *bags* of the tree-decomposition. Let $t \in \mathcal{T}$ have neighbours $\mathcal{N}(t) \subseteq \mathcal{T}$. The *torso* $\bar{\mathcal{V}}_t$ of $\mathcal{V}_t$ is the graph

$$G[\mathcal{V}_t] \cup \bigcup_{u \in \mathcal{N}(t)} K[\mathcal{V}_t \cap \mathcal{V}_u].$$

If $\mathcal{T}$ is a path then $(\mathcal{T}, \mathcal{V})$ is called a *path-decomposition*. The *width* of a tree-decomposition (or path-decomposition) is defined by $\max_{t \in T} |\mathcal{V}_t| - 1$.

Let $G = (V, E)$ and $H = (W, F)$ be graphs. We say that H is a *minor* of G, written $H \preceq G$, if there are disjoint connected nonempty subgraphs $(X_w)_{w \in W}$ in G such that for every edge $xy \in F$ there is an edge $ab \in E$ for some $a \in X_x$ and $b \in X_y$. We will need the following lemma:

Lemma 2.2. *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$, and let (A, B) be a separation of G with $|A \cap B| \leq k$ and such that $G[A \cap B]$ is a clique. If $K_{k+1} \preceq G$ then $K_{k+1} \preceq A$ or $K_{k+1} \preceq B$.*

Proof: Suppose $K_{k+1} \preceq G$ and let $X_1, \dots, X_{k+1}$ be disjoint nonempty connected subgraphs of G witnessing this (i.e., such that there is an edge in G between some $u \in X_i$ and $v \in X_j$ for every $1 \leq i < j \leq k+1$.) Since $|A \cap B| \leq k$, some X_i does not meet this set, and so must lie entirely on one side of the separation. Wlog we assume $X_1 \subseteq A \setminus B$.

Since there are no outgoing edges from X_1 to $B \setminus A$ we have $X_j \cap A \neq \emptyset$ for $j = 2, \dots, k+1$. Since $A \cap B$ is a clique we may replace X_j by $X_j \cap A$, thus $K_{k+1} \preceq A$. ■

Let $k \geq 0$ and let Σ be a surface with disjoint closed discs $D_1, \dots, D_k \subseteq \Sigma$. Up to homeomorphism, the space $\Sigma \setminus \bigcup_i \operatorname{int}(D_i)$ only depends on k and Σ (and not, say, on the positions of the discs D_i), and we denote it by $\Sigma - k$. For $i = 1, \dots, k$, let $f_i : [0,1] \to D_i$ be a path which follows the boundary curve C_i of D_i. Following Diestel [13], we call the C_i the *cuffs* of $\Sigma - k$, and $f_i(0) \in \Sigma$ the *root* of C_i. On each cuff C_i the path f_i induces a linear order on the points of C_i.

For a graph $G = (V, E)$, a $s \geq 0$ and a surface Σ, we say that G is *s-nearly embeddable into Σ* if there is a set $X \subseteq V$ with $|X| \leq s$ such that $G - X$ can be written as $H_0 \cup H_1 \cup \ldots \cup H_s$ so that

(a) there is an embedding Π of H_0 into $\Sigma - s$ such which meets cuffs only in vertices and which does not meet the root of a cuff,
(b) the (possibly empty) graphs $H_1, \dots, H_s$ are pairwise disjoint and common vertices of H_0 and H_i are exactly those vertices of H_0 which are mapped to C_i by Π,
(c) for $i = 1, \dots, s$, the graph H_i has a path-decomposition $(\mathcal{P}, \mathcal{V})$ of width $< s$ such that the vertices of $\mathcal{P}$ are the vertices of $H_0 \cap H_i$, ordered in the linear order of the

cuff C_i, and $v \in \mathcal{V}_v$ for these vertices. The graphs H_i are called *vortices* attached to H_0.

The vertices in X are called *apices* of the almost embedding.

III. k-Walks in Graphs with Excluded Minors

In this section we will prove the following lemma:

Lemma 3.3. *For every natural number r there is a k such that: If $G = (V, E)$ is a graph which does not contain a K_r-minor, then there is a supergraph $G' = (V, E')$ obtained from G by possibly adding edges such that G' does not contain a K_k-minor and there is a k-walk w through G'. Moreover, G' and w can be found in polynomial time for fixed r.*

Proof: Without loss of generality we assume that G is connected. We first compute a tree-decomposition $(\mathcal{T}, \mathcal{V})$ of G whose torsos are s-nearly embeddable into some surface into which K_r can not be embedded, for some s depending only on r. Such a decomposition exists by the Graph Structure Theorem (see, e.g., [13, Thm. 12.4.11]), and it can be computed in polynomial time for fixed r (cf. [27], [12]).

Assume that each bag $\mathcal{V}$ comes with an s-near embedding of its torso, i.e., for each bag $\mathcal{V}_t$ we are given a set Z_t of at most s apices, subgraphs $\mathcal{V}_t^{(0)}, \ldots, \mathcal{V}_t^{(s)}$ of $\bar{\mathcal{V}}_t \setminus Z_t$, and an embedding Π_t of the graph $\mathcal{V}_t^{(0)}$ into a surface into which K_r can not be embedded and such that $\mathcal{V}_t^{(1)}, \ldots, \mathcal{V}_t^{(s)}$ are attached as vortices to this embedding. The algorithm in [12] actually yields a decomposition and embeddings for which the tree $\mathcal{T}$ is rooted, say, with root t_r, and such that for every pair of nodes t and u such that u is a child of t we have

$$\mathcal{V}_t \cap \mathcal{V}_u \subseteq Z_u \tag{1}$$

and $(\mathcal{V}_t \cap \mathcal{V}_u) \setminus Z_t$ is either contained in a single bag of the path-decomposition of one $\mathcal{V}_t^{(i)}$ for $i \geq 1$ or is a set of size at most three and the vertices in this set lie on the boundary of a face of Π_t. By adding edges to G if necessary we may assume that

(D1) all bags are identical to their torsos, i.e., $\mathcal{V}_t = \bar{\mathcal{V}}_t$ for all $t \in \mathcal{T}$,

(D2) in every bag $t \in T$, all apices $z \in Z_t$ are connected to all other vertices in $\mathcal{V}_t$,

(D3) for every $t, u \in T$ such that u is a child of t, the set $(\mathcal{V}_t \cap \mathcal{V}_u) \setminus Z_t$ is either a clique contained in a single bag of the path-decomposition of one $\mathcal{V}_t^{(i)}$ for $i \geq 1$ or a face of Π_t of size three (i.e., a triangle).

We will need these properties later. After adding these edges the graph still excludes some clique as a minor; for a proof cf. [31, Thm. 1.1]. We will keep adding edges to G in the course of this proof. For ease of notation, we still call the resulting supergraphs G.

We add chords to all facial cycles of the embedding Π_t to turn it into a triangulation. In particular, the resulting graph is 3-connected (note that the neighbourhood of every $v \in \mathcal{V}_t^{(0)}$ induces a cycle as a subgraph, which is 2-connected), and since it is still embedded into the same surface as before we did not create a K_r-minor. By induction on the Euler genus eg of Π_t we show that there is a $2^{\mathrm{eg}+1}$-walk through $\mathcal{V}_t^{(0)}$ which can be computed in polynomial time.

The base case for the induction is $\mathrm{eg} = 0$, i.e., if $\mathcal{V}_t^{(0)}$ is a planar graph. In this case we invoke Gao and Richter's result (Theorem 2.1) to find a 2-walk. We extended $\mathcal{V}_t^{(0)}$ to a triangulation (which is 3-connected) exactly to be able to apply this theorem.

Otherwise there are non-contractible cycles in $\mathcal{V}_t^{(0)}$. We compute a shortest such cycle C; this can be done in polynomial time, cf. [42]. We need to distinguish two cases.

First assume C is two-sided and surface separating. Define left and right edges adjacent to C, as well as the left and right subgraph G_l and G_r, as in [38, Ch. 3]. Since C is surface separating, the graphs G_l and G_r are distinct, and by [38, Prop. 4.2.1] the Euler genus of Π_t equals the sum of the Euler genera of the induced embeddings Π_l and Π_r of $G_l \cup C$ and $G_r \cup C$. Since we assumed C to be non-contractible, neither of these graphs is planar, and so both have strictly positive Euler genus, which is strictly smaller than the Euler genus of Π_t. By induction, we find 2^{eg}-walks w_l and w_r in $G_l \cup C$ and $G_r \cup C$. Putting these together and joining them at some vertex in C yields a $2 \cdot 2^{\mathrm{eg}} = 2^{\mathrm{eg}+1}$-walk in $\mathcal{V}_t^{(0)}$.

Otherwise, if C is two-sided but not surface separating, or if C is one-sided, we cut along C as in [38, Lemma 4.2.4]. This results in a graph H whose Euler genus is lower than that of $\mathcal{V}_t^{(0)}$ and such that each vertex of C has two copies in H. Again, we use induction to find a 2^{eg}-walk in H, which directly gives us a $2 \cdot 2^{\mathrm{eg}} = 2^{\mathrm{eg}+1}$-walk in $\mathcal{V}_t^{(0)}$.

In order to extend this walk to a walk through $\mathcal{V}_t^{(0)} \cup \mathcal{V}_t^{(1)} \cup \ldots \cup \mathcal{V}_t^{(s)}$, we add edges to each $\mathcal{V}_t^{(i)}$, $i = 1, \ldots, s$, to turn each bag of the path-decomposition of $\mathcal{V}_t^{(i)}$ into a clique. Each bag contains some vertex $v \in \mathcal{V}_t^{(0)}$, and the first time the walk through $\mathcal{V}_t^{(0)}$ enters this v we make a detour through all nodes in its bag which have not been visited before, return to v, and then continue the walk. This results in a $(2^{\mathrm{eg}+1} + 1)$-walk through $\mathcal{V}_t^{(0)} \cup \mathcal{V}_t^{(1)} \cup \ldots \cup \mathcal{V}_t^{(s)}$.

So far we have found k'-walks through the torsos (excluding the apices) of our tree-decomposition, for some k' depending only on r. We added some edges, but the torsos still exclude some clique $K_{r'}$ as a minor: When we added chords to turn $\mathcal{V}_t^{(0)}$ into a triangulation, we also added chords through the cuffs at which the $\mathcal{V}_t^{(i)}$ are glued, but this can be done by connecting *one* node v_i on the cuff C_i to all other nodes on C_i, and by adding v to all bags of the path-decomposition of $\mathcal{V}_t^{(i)}$ we still get an almost-embeddable graph, which excludes some clique minor.

Furthermore, we only added edges within bags of our tree-decomposition $(\mathcal{T}, \mathcal{V})$, and these edges to not affect other bags, because we assumed all bags to be identical to their torsos: If $u, v \in V$ appear together in more than one bag, then there is already an edge between them in G. Thus $(\mathcal{T}, \mathcal{V})$ remains a tree-decomposition even after adding edges.

Now we extend these walks through the apices Z_t of each bag and paste the walks in the individual bags together to get a walk through the whole graph. We start at the root t_r of

$(\mathcal{T}, \mathcal{V})$ and pick, for every $z \in Z_{t_r}$, an arbitrary neighbour $v \in \mathcal{V}_{t_r}^{(0)}$ of z. If there is no such neighbour we add an edge to an arbitrary node. We make a detour through z the first time the walk visits v. This increases the number of times we visit v by one, but since $|Z_t| < s$, it can only turn our k'-walk into a $k_1 := (k' + s)$-walk.

We extend the k_1-walk step by step until it covers the whole graph G. Let $\tilde{\mathcal{T}}$ be the set of nodes of the tree-decomposition which are already covered, i.e., such that we already have a walk through $\bigcup_{t \in \tilde{\mathcal{T}}} \mathcal{V}_t$. By the last paragraph we may start with $\tilde{\mathcal{T}} = \{t_r\}$. Let $\partial\tilde{\mathcal{T}}$ be the set of nodes $t \in \tilde{\mathcal{T}}$ which still have children $u \in \mathcal{T} \setminus \tilde{\mathcal{T}}$. In each step we pick one $t \in \partial\tilde{\mathcal{T}}$ and extend the walk through all its children. The constructed walk will be a k_2-walk for some $k_2 > k_1$, but for every $t \in \partial\tilde{\mathcal{T}}$, no vertex from $\mathcal{V}_t \setminus Z_t$ will be visited more than k_1 times.

Let $t \in \partial\tilde{\mathcal{T}}$ be a bag with children $u_1, \dots, u_n \in \mathcal{T}$. We call

$$A_i := \mathcal{V}_{u_i} \cap \mathcal{V}_t$$

the *adhesion set* of u_1. The basic idea is to insert the k_1-walk w_i through the bag u_i into the walk w through t when the latter visits some vertex $v_i \in A_i$. However, inserting w_i at v_i increases the number of times we visit v_i by one, and since the number of children n is unbounded, we have to carefully choose the vertices $v_i \in A_i$ so that no single vertex is used more than a bounded number of times.

We first show that we may assume the adhesion sets A_i to be distinct. By Property (D3), each A_i is of the form

$$A_i = A_i^{(Z)} \cup \Delta_i,$$

where $A_i^{(Z)} := A_i \cap Z_t$ and Δ_i is a face of Π_t (the embedding of $\mathcal{V}_t^{(0)}$) or a clique which is contained in some bag of a path-decomposition of a vortex. It may well happen that an unbounded number of children of t have the same adhesion set, and we first show how to deal with this case.

Let u_1 be a child of t with adhesion set A, and let $u2, \dots, u_m$ be the other children of t with the same adhesion set. For each u_i we pick the endpoints $a_i, b_i \in \mathcal{V}_{u_1}^{(0)}$ of an arbitrary edge traversed from a_i to b_i by the walk w_i. (The case in which $\tilde{V}_{u_i}^{(0)}$ does not contain an edge can be dealt with easily.) We add edges $a_1b_2, a_2b_3, \dots, a_mb_1$ to G as in Figure 1. The resulting graph still excludes some clique, for if $r' \geq |A| + 3$ and none of the bags u_i has a $K_{r'}$ minor, then neither does $\bigcup_{i=1}^{m} \mathcal{V}_{u_i}$ with the added edges. This follows from Lemma 2.2, because $\mathcal{V}_{u_i}$ intersects $\bigcup_{j \neq i} \mathcal{V}_{u_j}$ in $A \cup \{a_i, b_i\}$, which is a clique of size $|A| + 2$.

We replace the bags $u_1, \dots, u_m$ by a single new bag $\tilde{u}$. While $\tilde{u}$ is no longer nearly embedded, it still excludes some clique minor (whose size depends only on r).

Using the new edges we connect the walks $w_1, \dots, w_m$ through the bags $u_i, \dots, u_m$ as follows: By our choice of a_i and b_i, each walk w_i traverses the edge a_ib_i at some point. We go from a_i to b_{i+1} instead (and from a_m to b_1). The resulting walk $\tilde{w}$ is a k_1-walk through $\tilde{u}$.

From now on we assume that no two children of t have identical adhesion sets. We want to pick a $v_i \in A_i$ such

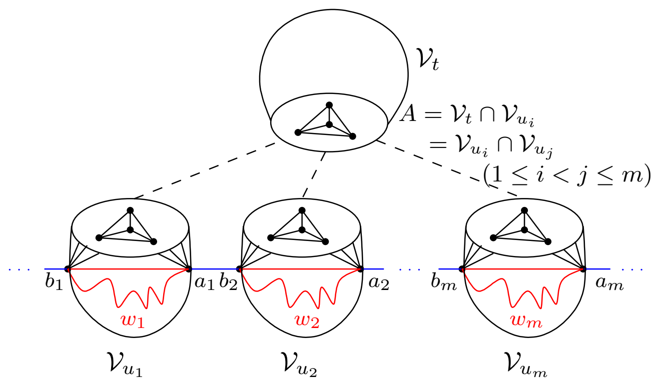

Fig. 1. The walks $w_1, \dots, w_m$ are joined by replacing edges a_ib_i on the walks (red) by edges a_ib_{i+1} (blue).

that no $v \in \mathcal{V}_t$ is chosen more than a bounded number of times. Having done so we can insert the walk w_i into the walk through t at v_i without increasing the number of times v_i is visited by too much.

We have to distinguish two cases: If Δ_i is contained in one bag of a path-decomposition of one of the vortices of t, we use the indexing vertex of that bag as v_i. By our bound on the path-width of the vertices, no v_i is used more than a bounded number of times.

For Δ_i which form a face of the embedding of $\mathcal{V}_t^{(0)}$ we proceed as follows: Let $\hat{G}$ be a barycentric subdivision of $(\mathcal{V}_t^{(0)}, \Pi_t)$, i.e., we introduce a new vertex v_F for every face F of Π_t and connect it to $v \in V(\mathcal{V}_t^{(0)})$ iff $v \in F$. Then $\hat{G}$ is again 3-connected and Π_t can be extended to an embedding of $\hat{G}$ in an obvious way. We compute a $2^{\text{eg}+1}$-walk $\hat{w}$ through $\hat{G}$ as above. Each v_F is visited at least once, and since all its neighbours are vertices of $\mathcal{V}_t^{(0)}$, its immediate predecessor on $\hat{w}$ when it is first visited is some $u_F \in \mathcal{V}_t^{(0)}$. Now if Δ_i is some face F of Π_t, we insert w_i into w when first visiting u_F. This way u_F is used at most $2^{\text{eg}+1}$ times. ■

IV. Defining a Successor Relation from a k-walk in FO

The following lemma allows us to interpret a successor relation from a k-walk in first-order logic.

Lemma 4.4. *Let σ be a finite relational vocabulary, A a finite σ-structure, and $w : [\ell] \to V(A)$ a k-walk through the Gaifman graph of A.*

Then there is a finite relational vocabulary σ_k and a first-order fomula $\varphi_{\text{succ}}^{(k)}(x, y)$, both depending only on k, and a $(\sigma \cup \sigma_k)$-expansion A' of A which can be computed from A and w in polynomial time, such that

- *The Gaifman-graphs of A' and A are the same,*
- *$\varphi_{\text{succ}}^{(k)}$ defines a successor relation on A'.*

Proof: We define a function $f : [\ell] \to [k]$ which counts how many times we have visited a vertex on the walk before, by

$$f(i) := |\{j \leq i \mid w(i) = w(j)\}| .$$

Furthermore, let $F : V(A) \to [k]$ count how many times we visit a vertex:

$$F(v) := |\{i \in [\ell] \mid w(i) = v\}| .$$

To simplify notation, we write $F(i)$ for $F(w(i))$ if $i \in [\ell]$.

We encode the k-walk w by binary relations E_{ab} with $a, b = 1, \dots, k$, in such a way that $(u, v) \in E_{ab}$ iff there is some $i \in [\ell - 1]$ such that

- $w(i) = u$ and $f(i) = a$, and
- $w(i + 1) = v$ and $f(i + 1) = b$.

I.e., after visiting u for the a-th time, the walk w proceeds to v, visiting it for the b-th time. Note that if $k = 1$ we can immediately define a successor relation by

$$\varphi^{(1)}_{\text{succ}}(x, y) := E_{11}xy.$$

If $k > 1$, we show how to interpret a $(k - 1)$-walk w' in first-order logic given a k-walk encoded by $\{E_{ab} \mid 1 \leq a, b \leq k\}$ as above. By daisy-chaining these interpretations we end up with a 1-walk (i.e., a Hamiltonian cycle). Plugging in the interpretation of this Hamiltonian cycle into $\varphi^{(1)}_{\text{succ}}$ defined above we obtain the formulas $\varphi^{(k)}_{\text{succ}}$.

In order to get from a k-walk to a $(k-1)$-walk, we look at all vertices which are visited k times, and "jump" over these vertices either when they are visited for the $(k - 1)$-th or for the k-th time. Jumping over a vertex can be done in first-order logic, but we must be careful to choose the vertices for jumping in such a way that we never jump over an unbounded number of vertices in a row, which is not possible in first-order logic. We encode the information on whether to jump when visiting for the $(k - 1)$-th or the k-th time in a new unary predicate P_k.

To be precise, let $\varphi_{k\text{-times}}(x)$ be a formula which states that x is visited k times:

$$\varphi_{k\text{-times}}(x) := \exists y \bigvee_{a=1}^{k} E_{ka}xy.$$

For those $u \in V(A)$ which are visited k-times, we agree to jump over them when they are visited for the k-th time if $u \in P_k$, and when they are visited for the $(k - 1)$-th time otherwise. Thus, if $w(i) = u$, $f(i) = k$ and $u \in P_k$, we want to remove the i-th step from the walk w and set

$$w_{-i}(j) := \begin{cases} w(j) & \text{if } j < i \\ w(j + 1) & \text{if } j \geq i \end{cases}$$

However, it may be the case that $w(i+1)$ is also visited k times and needs to be jumped over. We define first-order formulas which carry out a bounded number of such jumps as follows:

- For $a \in [k]$, the formula $\varphi_{\text{jump},a}(x)$ holds if we jump over x when visiting it for the a-th time:

$$\begin{aligned}\varphi_{\text{jump},1}(x), \dots, \varphi_{\text{jump},k-2}(x) &:= \bot \\ \varphi_{\text{jump},k-1}(x) &:= \varphi_{k\text{-times}}(x) \wedge \neg P_k x, \\ \varphi_{\text{jump},k}(x) &:= \varphi_{k\text{-times}}(x) \wedge P_k x.\end{aligned}$$

- For $r \geq 0$ and $a, b \in [k]$, the formula $\varphi^{(r)}_{\text{next},a,b}(x, y)$ holds if, when applying at most r consecutive jumps on entering x for the a-th time, we end up in node y which is visited for the b-th time in the (original) walk. Specifically:

$$\begin{aligned}\varphi^{(0)}_{\text{next},a,b}(x, y) &:= x \dot{=} y \wedge \delta_{ab} \\ \varphi^{(r+1)}_{\text{next},a,b}(x, y) &:= (\neg\varphi_{\text{jump},a}(x) \to (x \dot{=} y \wedge \delta_{ab})) \wedge \\ &\Big(\varphi_{\text{jump},a}(x) \to \exists z \bigvee_{c=1}^{k} \big(E_{ac}xz \wedge \varphi^{(r)}_{\text{next},c,b}(z, y)\big)\Big)\end{aligned}$$

Here, δ_{ab} is true if the indices a and b are the same:

$$\delta_{ab} := \begin{cases} \top & \text{if } a = b, \\ \bot & \text{otherwise.} \end{cases}$$

- We will show below how to choose the predicate P_k so that we never need to take more than two consecutive jumps. Thus, we can interpret a $(k - 1)$-walk w' using, for $a, b \in [k - 2]$, the formulas

$$\varphi_{E,a,b}(x, y) := \exists z \bigvee_{c=1}^{k} \Big(E_{ac}xz \wedge \varphi^{(2)}_{\text{next},c,b}(z, y)\Big) .$$

For $a \in [k - 2]$ we set

$$\varphi_{E,a,k-1}(x, y) := \\ \exists z \bigvee_{c=1}^{k} \Big(E_{ac}xz \wedge \big(\varphi^{(2)}_{\text{next},c,k-1}(z, y) \vee \varphi^{(2)}_{\text{next},c,k}(z, y)\big)\Big).$$

For $b \in [k - 2]$ we set

$$\begin{aligned}&\varphi_{E,k-1,b}(x, y) := \\ &\Big(\neg\varphi_{\text{jump},k-1}(x) \to \exists z \bigvee_{c=1}^{k} \big(E_{k-1,c}xz \wedge \varphi^{(2)}_{\text{next},c,b}(z, y)\big)\Big) \\ &\wedge \Big(\varphi_{\text{jump},k-1}(x) \to \exists z \bigvee_{c=1}^{k} \big(E_{k,c}xz \wedge \varphi^{(2)}_{\text{next},c,b}(z, y)\big)\Big)\end{aligned}$$

And finally

$$\begin{aligned}&\varphi_{E,k-1,k-1}(x, y) := \\ &\Big(\neg\varphi_{\text{jump},k-1}(x) \to \exists z \bigvee_{c=1}^{k} \Big(E_{k-1,c}xz \wedge \\ &\qquad \big(\varphi^{(2)}_{\text{next},c,k-1}(z, y) \vee \varphi^{(2)}_{\text{next},c,k}(z, y)\big)\Big)\Big) \wedge \\ &\Big(\varphi_{\text{jump},k-1}(x) \to \exists z \bigvee_{c=1}^{k} \Big(E_{k,c}xz \wedge \\ &\qquad \big(\varphi^{(2)}_{\text{next},c,k-1}(z, y) \vee \varphi^{(2)}_{\text{next},c,k}(z, y)\big)\Big)\Big)\end{aligned}$$

To define the predicate P_k, let $T \subseteq [\ell]$ be the set of indices $i \in [\ell]$ for which $F(i) = k$ and $f(i) \in \{k-1, k\}$. We obtain a perfect matching M on T by matching i and j iff $w(i) = w(j)$. We define a subset $J \subset [\ell]$ with the intended meaning that if

$i \in J$ we jump over the i-th step of w. The set J will satisfy the following two conditions:

- Every vertex v with $F(v) = k$ is jumped over exactly once, i.e.,
$$|\{i \in [\ell] \mid w(i) = v\} \cap J| = 1.$$
- We never jump more than twice in a row, i.e., if $i, i+1 \in J$ then $i + 2 \notin J$.

We partition the set $[\ell]$ into intervals of size 2, setting
$$U := \{\{1, 2\}, \{3, 4\}, \dots\},$$
with the last set $\{\ell\}$ being a singleton if ℓ is odd. Then the matching M defines a multigraph without loops on U, and the degree of $I \in U$ is at most 2. We direct the edges of M, viewed as edges in the multigraph (U, M), in such a way that every $I \in U$ has at most one incoming edge. The edges incident with I correspond to the elements of $I \cap T$, and we put $i \in I$ into J iff the edge corresponding to i is directed towards I. For every $k = 1, \dots, \lfloor(\ell - 1)/2\rfloor$ at most one of $2k - 1$ and $2k$ is in J, and therefore J satisfies the above requirements.

The definition of $P_k \subseteq V(G)$ is now straightforward:
$$\begin{aligned}P_k := \{v \in V(G) \mid F(v) = k \text{ and}\\ f(i) = k \text{ for the } i \in J \text{ with } w(i) = v\}.\end{aligned}$$
In summary, we end up with
$$\sigma_k := \{E_{ab} \mid a, b \in [k]\} \cup \{P_a \mid a = 2, \dots, k\},$$
and it is clear that our construction can be carried out in polynomial time. ■

V. Successor-Invariant FO on Minor-Closed Graph Classes

Now we can put the results from Sections III and IV together to prove our main result:

Theorem 5.5. *For every finite graph H, the model checking problem for successor-invariant first-order logic on the class of all finite structures whose Gaifman graph does not contain a H as a minor is fixed-parameter tractable when parameterised by the size of the formula.*

Proof: Since the minor-relation is transitive, and every finite graph is a minor of some clique K_r, it suffices to prove the theorem for cliques. Given a structure A whose Gaifman graph does not contain a K_r minor and a successor-invariant first-order formula φ, we first compute the Gaifman graph G of A. Using the algorithm of Lemma 3.3, we compute an s-walk $w : [\ell] \to V(A)$ for some s depending only on r such that after adding edges between successive vertices on the walk w the graph G still excludes a K_s minor.

We now apply Lemma 4.4 to expand A into a structure A' whose Gaifman graph contains no K_s minor and such that there is a first-order formula φ_{succ} which defines a successor relation in A'. We plug φ_{succ} into φ to obtain a formula φ' which satisfies
$$A \models \varphi \quad \Leftrightarrow \quad A' \models \varphi'.$$
Since φ_{succ} depends only on r, the size of φ' is bounded by a function of the size of φ, and since model checking for FO on classes of structures which exclude some minor was shown to be fixed-parameter tractable in [20], this proves our claim. ■

Conclusion

We proved that for every $r \in \mathbb{N}$, the model checking problem for successor-invariant first-order logic on the class of structures whose Gaifman graph does not contain a clique of size r as a minor is fixed-parameter tractable when parameterised by the size of the input formula. This comes close to corresponding results for first-order logic.

The stronger tractability result for first-order logic obtained by Dawar et al. in [11] and by Dvořák et al. in [15] do not rely on decompositions into almost embeddable parts and instead use weaker decompositions which are easier to compute. Since our proof heavily relies on the topological properties of almost embeddable graphs, it is not at all clear how our result for successor-invariant FO could be strengthened to obtain similar results as for FO, and this calls for further research.

Another well-studied logic similar to successor-invariant FO is order-invariant first-order logic. By a result of Grohe et al. [28], order-invariant FO enjoys locality properties similar to those of FO, and these locality properties lie at the heart of all tractability results for FO. Whether they can be used to obtain tractability results for order-invariant FO is an interesting open question.

References

[1] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[2] Michael Benedikt and Luc Segoufin. Regular tree languages definable in fo. In *STACS*, pages 327–339, 2005.

[3] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. of Mathematical Foundations of Computer Science (MFCS)*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36, 1997.

[4] Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science*, volume 4271 of *LNCS*, pages 1–14. Springer, 2006.

[5] John M. Boyer and Wendy J. Myrvold. On the cutting edge: Simplified O(n) planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.

[6] Julius R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

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

[8] Bruno Courcelle. Canonical graph decompositions. Presentation given at a workhop on Graph Structure and MSO, Bordeaux, see http://www.labri.fr/perso/courcell/Conferences/ExpoCanDecsJuin2012.pdf, 2012.

[9] Bruno Courcelle, Johann Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[10] Bruno Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science (TCS)*, 109(1-2):49–82, 1993.

[11] Anuj Dawar, Martin Grohe, and Stephan Kreutzer. Locally excluding a minor. In *Logic in Computer Science (LICS)*, pages 270–279, 2007.

[12] Erik D. Demaine, Mohammad Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation and coloring. In *FOCS*, 2005.

[13] Reinhard Diestel. *Graph Theory*. Number 173 in GTM. Springer, 4th edition, 2012.

[14] Rod Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1998.

[15] Zdeněk Dvořák, Daniel Král, and Robin Thomas. Deciding first-order properties for sparse graphs. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[16] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, 2nd edition, 1999.

[17] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Springer, 2nd edition, 1994.

[18] Viktor Engelmann, Stephan Kreutzer, and Sebastian Siebertz. First-order and monadic second-order model-checking on ordered structures. In *Logics in Computer Science*, pages 275–284, 2012.

[19] Jörg Flum, Markus Frick, and Martin Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002.

[20] Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001.

[21] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. ISBN 3-54-029952-1.

[22] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48:1148 – 1206, 2001.

[23] Zhicheng Gao and R. Bruce Richter. 2-walks in circuit graphs. *J. Comb. Theory, Ser. B*, 62(2):259–267, 1994.

[24] Zhicheng Gao, R. Bruce Richter, and Xingxing Yu. 2-walks in 3-connected planar graphs. *Australasian Journal of Combinatorics*, 11:117–122, March 1995.

[25] Martin Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004.

[26] Martin Grohe. Logic, graphs, and algorithms. In E.Grädel T.Wilke J.Flum, editor, *Logic and Automata – History and Perspectives*. Amsterdam University Press, 2007.

[27] Martin Grohe, Ken-ichi Kawarabayashi, and Bruce Reed. A simple algorithm for the graph minor decomposition. In *SODA*, 2013.

[28] Martin Grohe and Thomas Schwentick. Locality of order-invariant first-order formulas. *ACM Trans. Comput. Logic*, 1(1):112–130, July 2000.

[29] Petr Hlinený and Sang il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008.

[30] John Hopcroft and Robert E. Tarjan. Efficient planarity testing. *JACM*, 21(4):549–568, 1974.

[31] Gwenaël Joret and David R. Wood. Complete graph minors and the graph minor structure theorem. *Journal of Combinatorial Theory, Series B*, 103(1):61–74, 2013.

[32] Joachim Kneis, Alexander Langer, and Peter Rossmanith. Courcelle's theorem - a game-theoretic approach. *Discrete Optimization*, 8(4):568–594, 2011. Preprint available at http://tcs.rwth-aachen.de/ langer/.

[33] Stephan Kreutzer. On the parameterised intractability of monadic second-order logic. In *Proc. of Computer Science Logic (CSL)*, 2009.

[34] Stephan Kreutzer. Algorithmic meta-theorems. In Javier Esparza, Christian Michaux, and Charles Steinhorn, editors, *Finite and Algorithmic Model Theory*, London Mathematical Society Lecture Note Series, chapter 5, pages 177–270. Cambridge University Press, 2011. a preliminary version is available at Electronic Colloquium on Computational Complexity (ECCC), TR09-147, http://www.eccc.uni-trier.de/report/2009/147.

[35] Stephan Kreutzer and Siamak Tazari. Lower bounds for the complexity of monadic second-order logic. In *Logic in Computer Science (LICS)*, 2010.

[36] Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Sequoia homepage. http://sequoia.informatik.rwth-aachen.de/sequoia, 2012.

[37] Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. Spinger-Verlag, 2004.

[38] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[39] Hannu Niemistö. On locality and uniform reduction. In *Logic in Computer Science (LICS)*, pages 41–50, 2005.

[40] Benjamin Rossman. Successor-invariant first-order logic on finite structures. *J. Symb. Log.*, 72(2):601–618, 2007.

[41] Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 5:505–526, 1996.

[42] Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Comb. Theory, Series B*, 48:155–177, 1990.

[43] William T. Tutte. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2001.