Lower Bounds for the Complexity of Monadic Second-Order Logic

Stephan Kreutzer
University of Oxford
stephan.kreutzer@comlab.ox.ac.uk

Siamak Tazari
Humboldt Universität zu Berlin
tazari@informatik.hu-berlin.de

***Abstract*—Courcelle's famous theorem from 1990 states that any property of graphs definable in monadic second-order logic $(\mathbf{MSO_2})$ can be decided in linear time on any class of graphs of bounded tree-width, or in other words, $\mathbf{MSO_2}$ is fixed-parameter tractable in linear time on any such class of graphs. From a logical perspective, Courcelle's theorem establishes a sufficient condition, or an upper bound, for tractability of $\mathbf{MSO_2}$-model checking.**

Whereas such upper bounds on the complexity of logics have received significant attention in the literature, almost nothing is known about corresponding lower bounds. In this paper we estbalish a strong lower bound for the complexity of monadic second-order logic. In particular, we show that if $\mathcal{C}$ is any class of graphs which is closed under taking sub-graphs and whose tree-width is not bounded by a poly-logarithmic function (in fact, $\log^c n$ for some small c suffices) then $\mathbf{MSO_2}$-model checking is intractable on $\mathcal{C}$ (under a suitable assumption from complexity theory).

I. Introduction

In 1990, Courcelle proved a fundamental result stating that every property of graphs definable in *monadic second-order logic with edge set quantification* (MSO_2), the extension of first-order logic by quantification over sets of vertices and edges, can be decided in linear time on any class $\mathcal{C}$ of graphs of bounded tree-width. This theorem has important consequences both in logic and in algorithm theory. In the theory of efficient algorithms on graphs, it can often be used as a simple way of establishing that a property can be solved in linear time on graph classes of bounded tree-width. Besides being of interest for specific algorithmic problems, results such as Courcelle's and similar *algorithmic meta-theorems* lead to a better understanding how far certain algorithmic techniques, such as dynamic programming on bounded tree-width graphs, range and establish general upper bounds for the parameterized complexity of a wide range of problems. See [4], [7] for recent surveys on algorithmic meta-theorems.

From a logical perspective, Courcelle's theorem establishes a sufficient condition for tractability of MSO_2 formula evaluation on classes of graphs or structures: whatever the class $\mathcal{C}$ may look like, if it has bounded tree-width, then MSO_2-model checking is tractable on $\mathcal{C}$. An obvious question is how tight the theorem actually is, i.e. whether it can be extended to classes of unbounded tree-width and if so, how large the tree-width of graphs in the class can be in general. Given the considerable interest in Courcelle's theorem, and the far-reaching consequences that extensions of this result to interesting classes of graphs of unbounded tree-width would have, it is surprising that not much is known about such limits for MSO_2-model checking. To fully understand the (parameterized) complexity of monadic second-order logic with respect to particular classes of graphs, we need to understand necessary conditions for tractability as much as sufficient conditions; but for some reason necessary conditions have so far not been studied in much depth.

A first lower bound for the complexity of monadic second-order logic appeared in [8] and has been extended in [6]. In these papers, it was shown that MSO_2-model-checking is not fixed-parameter tractable on any class of graphs where a) the tree-width is strongly unbounded by $\log^{28} n$ (see Definition 1.1 below) and b) which are *closed under colourings* for a fixed set Γ of colours, i.e. if $G \in \mathcal{C}$ and G' is obtained from G by colouring some vertices or edges by colours from Γ, then $G' \in \mathcal{C}$. These papers establish powerful logical and algorithmic tools for proving such intractability results and we will resort to some of these tools below. However, closure under colourings is a very strong condition as it allows to "mark" bad sub-structures in a graph.

In this paper we aim for a much stronger intractability result for MSO_2. To state the main result formally we first need some notation.

Definition 1.1. *The tree-width of a class $\mathcal{C}$ of graphs is* strongly unbounded *by a function $f : \mathbb{N} \to \mathbb{N}$ if there is $\varepsilon < 1$ and a polynomial $p(x)$ s.t. for all $n \in \mathbb{N}$ there is a graph $G_n \in \mathcal{C}$ with*

1) *the tree-width of G_n is between n and $p(n)$ and is not bounded by $f(|G_n|)$ and*
2) *given n, G_n can be constructed in time 2^{n^ε}.*

The degree of the polynomial p is called the gap-degree *of $\mathcal{C}$ (with respect to f). The tree-width of $\mathcal{C}$ is* strongly unbounded poly-logarithmically *if it is strongly unbounded by $\log^c n$, for all $c \geq 1$.*

The main result of the paper is the following theorem.

Theorem 1.2. *Let $\mathcal{C}$ be a class of graphs closed under sub-graphs, i.e. $G \in \mathcal{C}$ and $H \subseteq G$ implies $H \in \mathcal{C}$.*

1) *If the tree-width of $\mathcal{C}$ is strongly unbounded by $\log^{28\gamma} n$, where $\gamma > 1$ is larger than the gap-degree of $\mathcal{C}$, then $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C})$ is not in XP, and hence not fixed-parameter tractable, unless* SAT *can be solved in sub-exponential time $2^{o(n)}$.*
2) *If the tree-width of $\mathcal{C}$ is strongly unbounded poly-logarithmically then $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C})$ is not in XP unless all problems in the polynomial-time hierarchy can be solved in sub-exponential time.*

Here, $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C})$ refers to the parameterized model-checking problem for MSO_2 as defined below. We will give a justification for the two conditions in Definition 1.1 below, once we have sketched the main arguments of the proof. To give an example, the theorem implies that the class $\mathcal{C}$ of all (or all planar, bipartite, etc.) graphs G of tree-width $\mathrm{tw}(G) \leq \log^{29} |G|$ does not have tractable MSO_2 model-checking unless SAT can be solved in sub-exponential time.

Related work. The result in this paper complements the intractability result in [6] in that it refers to classes of graphs closed under sub-graphs and does not require any colours, a much more natural condition.

In [4, Conjecture 8.3], Grohe conjectures the following. (The original conjecture is formulated in terms of branch-width but this is equivalent to the formulation here.)

Conjecture (Grohe). *Let $\mathcal{C}$ be a class of graphs that is closed under taking sub-graphs. Suppose that the tree-width of $\mathcal{C}$ is not poly-logarithmically bounded, that is, there is no constant c such that $\mathrm{tw}(G) \leq \log^c |G|$ for every $G \in \mathcal{C}$. Then the model-checking problem of MSO_2 is not fixed parameter tractable on $\mathcal{C}$.*

Clearly, with current technology there is no hope to prove any such conjecture without relating it to assumptions in complexity theory (as the conjecture implies P $\neq$ PSPACE). In this sense our result only proves Grohe's conjecture modulo complexity theoretical assumptions and the additional conditions on strongly unboundedness necessitated by this. On the other hand, our result is stronger than the conjecture in that we only require a fixed log-power rather than polylog.

In [10], Makowsky and Mariño study similar questions in relation to classes of graphs closed under topological minors (see below). They show that any such class must have bounded tree-width for MSO_2 model-checking to be fpt. Closure under topological minors is a much stronger condition simplifying the proof significantly. However, in the same paper, the authors give examples for classes of graphs of unbounded clique-width but with tractable MSO_1 model-checking. These examples can be adapted to examples of classes of graphs which are closed under sub-graphs, whose tree-width is only bounded logarithmically (but which almost have logarithmic tree-width) and on which MSO_2 model-checking is tractable. This shows that in full generality, our results can not be strengthend much beyond the $\log^{28\gamma} n$ bound postulated in our result.

Overview of the proof. Let us briefly sketch the main idea of the proof. Let $\mathcal{C}$ be a class of graphs with tree-width strongly unbounded by $\log^c n$, for some suitable c.

We aim at reducing the propositional satisfiability problem SAT to $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C})$. Towards this aim we will first construct an MSO_2-formula φ, depending only on a Turing-machine deciding SAT, and then, given a SAT-instance w, construct a graph $G_w \in \mathcal{C}$ so that $G_w \models \varphi$ if, and only if, w is satisfiable. The idea is to encode the instance w in the graph G_w so that a) the instance can be decoded by the MSO_2-formula φ and b) the graph G_w contains enough structure so that the formula φ can simulate the run of a Turing-machine deciding SAT on input w.

Similar ideas in connection with tree-width have been employed in the past and the usual approach is to use the result, known as the excluded grid theorem (see e.g. [12]), that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that every graph of tree-width $f(k)$ contains a $k \times k$-grid (as a minor, which is good enough). Such a grid provides the structure we need to simulate runs of Turing-machines in MSO_2 and encoding the SAT instance w in a grid can easily be done by deleting certain edges (see Section V-C).

However, the best known bound for the function f known to date is exponential and as we are dealing with graphs of tree-width only logarithmic in the number of vertices, the grids we are guaranteed to find in this way are essentially only of order $\log \log |G_w|$ which is much too small for any reduction to work.

Instead of using grids, therefore, we will use a new structural characterisation of tree-width developed by Reed and Wood [11] and made algorithmic in [6] which replaces grids by *grid-like minors*. It was shown in [11] that any graph contains a grid-like minor of order polynomial in its tree-width and in [6] it was shown that these are computable in polynomial time. The main problem with grid-like minors is that a) they can resemble cliques rather than grids and b) they do not occur as minors of the graph itself but only of the intersection graph of sets of pairwise disjoint paths (see Section II for details). As indicated above, we would like to encode a SAT-instance w in a grid by deleting certain edges. But as grid-like minors only occur as minors of intersection graphs, deleting an edge in a graph G has no predictable implication for the grid-like minor which makes encoding SAT-instances using grid-like minors extremely difficult.

Therefore, instead of encoding SAT-instances in grid-like minors directly, we will impose a labelling of the grid-like minor externally. For this, given a SAT-instance $w := w_1 \dots w_l$ and a graph G of sufficiently high tree-width, we construct a tree $T \subseteq G$ which has a special structure so that there is an MSO_2-formula defining a linear order on trees of this structure. Furthermore, this particular structure of the tree allows us to encode the letters w_i in sub-trees of T containing some of the leaves (we will call these *single crosses* (encoding 0) and *double crosses* (encoding

1)). Hence, the order imposed on T together with the ability to encode letters allows us to encode the SAT-instance w in T. We will then show that G also contains a grid-like minor which is attached to the tree T so that the word encoded in T can be transferred to a unique labelling of the grid-like minor. Hence, we will use this external tree to encode the SAT-instance and the grid-like minor as the structure we need to simulate the run of a Turing-machine on the encoded input. The tree T together with the grid-like minor attached to it is called a *labelled tree-ordered web* and is illustrated in Figure 2.

Finally, as we assume that the class $\mathcal{C}$ of graphs we work in is closed under sub-graphs, this labelled tree-ordered web occurs as a graph in $\mathcal{C}$. Hence, if evaluating the MSO_2-formula which decodes the encoded SAT-instance and simulates the run of a TM on it was fixed-parameter tractable, then we could solve SAT in sub-exponential time.

On strongly unbounded tree-width. Let us give some justification for the two conditions in Definition 1.1. The first condition is a consequence of the fact that we prove our main result by reducing an NP-hard problem to $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C})$. Without this condition there could simply be too few graphs of high tree-width in $\mathcal{C}$ to define a reduction. To give an example, fix a constant c and let H_n be the graph constructed from the $n \times n$-grid by replacing every edge by a path on $\frac{2^{\sqrt[c]{n}}}{m}$ vertices, where $m = n^2$. The resulting graph has $\mathcal{O}(2^{\sqrt[c]{n}})$ vertices and tree-width n. Now let $\mathcal{C}' := \{H_n : n = 2^{2^i}, i > 0\}$ and let $\mathcal{C}$ be the sub-graph closure of $\mathcal{C}'$. If $c > 29$, then the tree-width of $\mathcal{C}$ is unbounded by $\log^{29} n$ but not strongly unbounded by this function, while being closed under taking subgraphs. To see this, take a graph $H_n \in \mathcal{C}'$, for some $n = 2^{2^i}$, $i > 2$. Any sub-graph $H \subseteq H_n$ is either acyclic, and therefore has tree-width 1, or it contains a path of length $\frac{2^{\sqrt[c]{n}}}{m}$. Thus, H_n does not contain any sub-graph $H \subseteq H_n$ of tree-width $2^i \leq \mathrm{tw}(H) \leq p(2^i)$ such that $\mathrm{tw}(H) > \log^c |H|$, for any fixed polynomial p. It follows that if we wanted to use $\mathcal{C}$ for a reduction as outlined above, there wouldn't be enough graphs of large tree-width to reduce to: given an instance of SAT of length 2^i for an i that is not close to a power of 2, we would have no chance in identifying a graph in $\mathcal{C}$ to perform a reduction in polynomial time. Therefore, as long as we have to rely on reductions to prove results as in this paper, a condition similar to Condition 1 seems necessary. The second condition is necessary to prevent artificial cases where constructing a graph in the class $\mathcal{C}$ is already so expensive that any reduction would take too much time.

Organisation. The paper is organised as follows. In Section II we recall notation and concepts from graph theory. We recall monadic second-order logic in Section III. In Section IV we define the labelled tree-ordered webs discussed above and show that every graph of sufficient tree-width contains such a structure. This is the main technical part of the paper. Finally, in Section V we present the logical aspects of the paper. We conclude in Section VI.

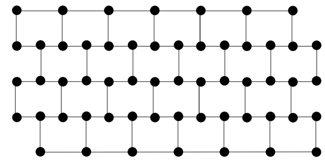

Figure 1. Elementary 4×6-wall.

II. Preliminaries

We use standard notation from graph theory and refer to [2] for details. All graphs in this paper are simple and undirected. We write $V(G)$ and $E(G)$ for the set of vertices and edges of a graph G and assume $V(G) \cap E(G) = \varnothing$. We let $\deg_G(v)$ denote the degree of vertex v in G. A *path* $P \subseteq G$ in a graph G is a connected acyclic sub-graph in which $\deg_P(v) \leq 2$ for every $v \in V(P)$.

Tree-width is a global connectivity measure of graphs that was introduced by Robertson and Seymour in their graph minor series. Essentially, it associates to each graph G a number $\mathrm{tw}(G) \in \mathbb{N}$ measuring how similar a graph is to being a tree. We will not need the precise definition in this paper and therefore refer the reader to [2] for a definition of tree-width.

Definition 2.1. *Let $f : \mathbb{N} \to \mathbb{N}$ be a function and $\mathcal{C}$ be a class of graphs. The tree-width of $\mathcal{C}$ is* bounded *by f if $\mathrm{tw}(G) \leq f(|G|)$ for all $G \in \mathcal{C}$. $\mathcal{C}$ has* bounded tree-width *if its tree-width is bounded by a constant.*

Many natural classes of graphs, for instance series-parallel graphs, are found to have bounded tree-width.

Definition 2.2. *A* sub-division *of a graph H is a graph H' obtained from H by iteratively replacing some edges by paths of length 2. The original vertices of H in H' are called the* nails *of H in H'. If a graph G contains a sub-division of H, we call H a* topological minor *of G.*

Definition 2.3. *Let $n, m > 0$ be integers. An* elementary $n \times m$-wall *is a graph as indicated in Figure 1. An* $n \times m$-wall *is a sub-division of an elementary $n \times m$-wall. The* nails *of a wall are the vertices of the elementary wall it is obtained from by sub-dividing edges.*

We will always think of the vertices of a wall as being numbered in a way that $(1, 1)$ is the vertex in the "bottom-left corner". The "bottom-row" of an $n \times m$-matrix is then the row 1.

Let $\mathcal{P}$ and $\mathcal{Q}$ each be a set of disjoint paths of a graph G. We denote by $\mathcal{I}(\mathcal{P}, \mathcal{Q})$ the *intersection graph* of $\mathcal{P}$ and $\mathcal{Q}$ defined as the bipartite graph with vertex set $\mathcal{P} \cup \mathcal{Q}$ and an edge between two vertices if and only if the corresponding paths intersect. The following definition is adapted from Reed and Wood's [11] definition of a *grid-like minor*:

Definition 2.4. *Let $\mathcal{P}$ and $\mathcal{Q}$ be each a set of disjoint paths in a graph G. $(\mathcal{P}, \mathcal{Q})$ is called a* topological grid-like minor of order ℓ *in G if $\mathcal{I}(\mathcal{P}, \mathcal{Q})$ contains a sub-division of the complete graph K_ℓ. The* nails *of $(\mathcal{P}, \mathcal{Q})$ are the paths corresponding to the nails of the sub-division of K_ℓ in $\mathcal{I}(\mathcal{P}, \mathcal{Q})$.*

III. Monadic Second-Order Logic

In this section we briefly recall the definition of monadic second-order logic. As we are mainly interested in graphs we only introduce MSO_2 on graphs.

The class of formulas of *monadic second-order logic with edge set quantification*, denoted MSO_2, is defined as the extension of first-order logic by quantification over sets of edges and sets of vertices. However, for the purpose of this paper it is more convenient to define it formally as monadic second-order logic on *incidence structures*.

Signatures and Structures. A signature σ is a finite set of relation symbols R each of arity *ar*(R). A σ-structure A consists of a *universe* $U(A)$ and for each $R \in \sigma$ an *ar*(R)-ary relation $R(A) \subseteq (U(A))^{ar(R)}$.

Incidence Structures. The signature σ_{graph} of incidence structures is defined as $\sigma_{graph} := \{V, E, \in\}$, where V, E are unary and $\in$ is a binary relation symbol. We will always use $\in$ in infix notation and write $v \in^A e$ instead of $(v, e) \in \in(A)$. With any graph G we associate a σ_{graph}-structure $A := \mathcal{G}(G)$, its *incidence structure*, with universe $U(A) := V(G)\dot\cup E(G)$ and $V(A) := V(G)$, $E(A) := E(G)$ and $v \in^A e$ if $v \in V(G), e \in E(G)$ and v and e are incident in G. We will not usually distinguish between a graph G and its incidence structure.

Monadic Second-Order Logic (MSO_2). MSO_2 is the extension of first-order logic by quantification over sets of elements (which can be vertices or edges). That is, in addition to first-order variables, which we will denote by small letters $x, y, ...$, there are variables $X, Y, ...$ ranging over sets of elements. Formulas of $\text{MSO}_2[\sigma]$ are then build up inductively by the rules for first-order logic $\text{FO}[\sigma]$ with the following additional rules: if X is a second-order variable and $\varphi \in \text{MSO}_2[\sigma\dot\cup\{X\}]$, then $\exists X\varphi \in \text{MSO}_2[\sigma]$ and $\forall X\varphi \in \text{MSO}_2[\sigma]$ with the obvious semantics where, e.g., a formula $\exists X\varphi$ is true in a σ-structure G if there is a subset $X' \subseteq U(G)$ such that φ is true in G if the variable X is interpreted by X'. We write $G \models \varphi$ to indicate that φ is true in G.

If $\varphi(x)$ is a formula with a free first-order variable x and G is a structure, we write $\varphi(G)$ for the set $\{v \in U(G) : G \models \varphi[v]\}$. See [9] for more on MSO_2.

To give an example, the following MSO_2-formula

$$\exists C_1 \exists C_2 \exists C_3 \forall x \bigvee_{i=1}^{3} x \in C_i \wedge \\ \forall x \forall y \big((x, y) \in E \rightarrow \bigwedge_{1 \leq i \leq 3} \neg(x \in C_i \wedge y \in C_i)\big)$$

is true in a graph G if, and only if, G is 3-colourable.

Model Checking. The *model checking problem* $\text{MC}(\text{MSO}_2)$ for MSO_2 is defined as the problem, given a structure G and a formula $\varphi \in \text{MSO}_2$, to decide if $G \models \varphi$. In [13], Vardi proved that $\text{MC}(\text{MSO}_2)$ is PSPACE-complete. However the hardness result crucially uses the fact that the formula is part of the input (and in fact holds on a fixed two-element structure), whereas we are primarily interested in the complexity of checking a fixed formula expressing a graph property in a given input graph. We therefore study model-checking problems in the framework of *parameterized complexity* (see [3] for background on parameterized complexity).

Definition 3.1. *Let $\mathcal{C}$ be a class of σ-structures. The* parameterized model-checking problem $\text{MC}(\text{MSO}_2, \mathcal{C})$ *for* MSO_2 *on $\mathcal{C}$ is defined as the problem to decide, given $G \in \mathcal{C}$ and $\varphi \in \text{MSO}_2[\sigma]$, if $G \models \varphi$. The* parameter *is $k := |\varphi|$.*

$\text{MC}(\text{MSO}_2, \mathcal{C})$ is *fixed-parameter tractable* (fpt), if for all $G \in \mathcal{C}$ and $\varphi \in \text{MSO}_2[\sigma]$, $G \models \varphi$ can be decided in time $f(k) \cdot |G|^c$, for some computable function f and $c \in \mathbb{N}$, where $k := |\varphi|$ is the parameter. The problem is in the class XP, if it can be decided in time $|G|^{f(k)}$.

An important aspect of parameterized complexity is that it is invariant under syntactic variations of the logic, i.e. if $\mathcal{L}$ and $\mathcal{L}'$ are equivalent in the sense that formulas of one logic can effectively be translated into equivalent formulas of the other logic, then $\mathcal{L}$ is fpt on a class $\mathcal{C}$ if, and only if, $\mathcal{L}'$ is fpt on $\mathcal{C}$. The corresponding statement is false for classical complexity.

As, for instance, the NP-complete problem 3-Colourability is definable in MSO_2, $\text{MC}(\text{MSO}_2, \text{GRAPHS})$, the model-checking problem for MSO_2 on the class of all graphs, is not fixed-parameter tractable unless $P = \text{NP}$. However, Courcelle proved that if we restrict the class of admissible input graphs, then we can obtain much better results.

Theorem 3.2 ([1]). $\text{MC}(\text{MSO}_2, \mathcal{C})$ *is fixed-parameter tractable, with parameter $|\varphi| + \text{tw}(G)$, on any class $\mathcal{C}$ of graphs of tree-width bounded by a constant.*

IV. Labelled Tree-Ordered Webs

The goal of this section is to prove the main algorithmic aspects of this paper. As indicated in the introduction, we aim at encoding instances w of an NP-hard problem P, i.e. w is a word over the alphabet $\{0, 1\}$, in graphs of large enough tree-width. The core algorithmic problem is to identify a structure so that there is a polynomial p such that given a word w of length m and a graph of tree-width at least $p(m)$, G contains such a structure encoding w as a sub-graph. The structure we are after, which we call *labelled tree-ordered webs*, is indicated in Figure 2. Starting with a certain graph provided in [6] that contains a grid-like minor of large order,

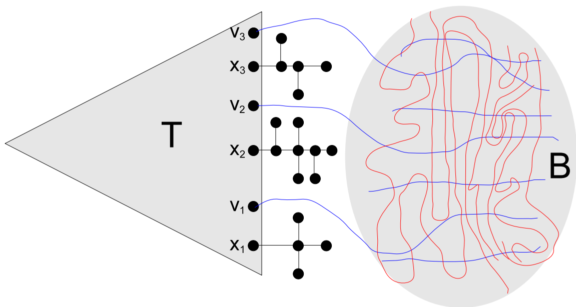

Figure 2. A labelled tree-ordered web encoding `010`.

we incrementally modify it in the sub-sections below, adding more and more structure to it, until we obtain a sub-graph encoding w in a way that is recognisable by an MSO_2-formula; and once this is done we can guess in MSO_2 an accepting run of a Turing-machine deciding the problem P on input w. Due to lack of space, some proofs are omitted in this extended abstract.

A. Tree-Webs

Definition 4.1. *A tree T is* sub-cubic *if its maximum degree is at most 3. A set $A \subseteq V(T)$ is called* flat *if every vertex $v \in A$ has degree at most 2 in T.*

The notion of a *tree-web*, defined below, is central to this work; in the subsequent sections, we will refine this notion to finally obtain the structure that we need.

Definition 4.2. *A* tree-web *of order ℓ is a tuple $\mathcal{W} = (G, T, A, \mathcal{P}, \mathcal{Q})$, so that*

1) *T is a sub-cubic tree,*
2) *$(\mathcal{P}, \mathcal{Q})$ is a topological grid-like minor of order ℓ^2 whose nails are paths from $\mathcal{P}$,*
3) *$G = T \cup \bigcup \mathcal{P} \cup \bigcup \mathcal{Q}$ is a graph of maximum degree 4,*
4) *T only intersects with nails of $(\mathcal{P}, \mathcal{Q})$,*
5) *the paths from $\mathcal{P}$ that are nails are either disjoint from T or intersect T in exactly one endpoint, and*
6) *$A = V(T) \cap V(\bigcup \mathcal{P})$ is flat in T.*

The vertices of A are called the good *vertices of $\mathcal{W}$. The paths in $\mathcal{P}$ that start at a vertex in A are called* good paths.

In case *all* the nails in $\mathcal{P}$ are good and hence, intersect with T, i.e. $|A| = \ell^2$, we call the structure a *full tree-web*. The following lemma is an almost immediate consequence of the main algorithmic result in [6].

Lemma 4.3. *There is a constant c and a polynomial-time algorithm that given a graph G of tree-width at least $c\ell^{14}$ finds either an $\ell \times \ell$-wall or a full tree-web of order ℓ in G.*

Definition 4.4. *A sub-tree-web of a tree-web $\mathcal{W} = (G, T, A, \mathcal{P}, \mathcal{Q})$ is a tree-web $\mathcal{W}' = (G', T', A', \mathcal{P}', \mathcal{Q}')$ with $G' \subseteq G$. In this case, we write $\mathcal{W}' \subseteq \mathcal{W}$.*

Definition 4.5. *A* full sub-tree *of a tree T is a connected component of $T - e$, for some edge $e \in E(T)$. A tree-web $\mathcal{W} = (G, T, A, \mathcal{P}, \mathcal{Q})$ is* nice *if*

1) *G has no vertex of degree 1,*
2) *if $P = (v_0, \ldots, v_k)$ is a good path with $v_0 \in A$, then v_1 does not lie on any other path,*
3) *every full sub-tree of T with at least 2 vertices contains at least 2 good vertices.*
4) *every leaf of T is good, and*
5) *the neighbour of every leaf of T in T is good.*

Note that the last two conditions are implied by the third one. The proof of Lemma 4.7 below is based on the combinatorial Lemma 4.6. They imply that we may assume w.l.o.g. that any given tree-web is nice.

Lemma 4.6. *Let $G := K_k$ be a clique on k vertices and assume at most k edges of G are coloured red and the rest are coloured blue. Then G contains a blue clique H of size $\lfloor k/3 \rfloor$ that can be found in polynomial time.*

Lemma 4.7. *Given a tree-web $\mathcal{W}$ of order ℓ, one can construct a nice tree-web $\mathcal{W}'$ of order at least $\lfloor \ell/3 \rfloor$ with $\mathcal{W}' \subseteq \mathcal{W}$ in polynomial time.*

Next, we would like to identify a unique root for a tree-web:

Definition 4.8. *A tree-web $(G, T, A, \mathcal{P}, \mathcal{Q})$* admits a definable root *if it contains exactly one vertex $r \in V(T)$ of degree 3 such that two components of $G - r$ are single vertices and the third contains at least one edge.*

Lemma 4.9. *Given a nice tree-web $\mathcal{W} = (G, T, A, \mathcal{P}, \mathcal{Q})$ with at least 3 good vertices, one can construct a sub-tree-web $\mathcal{W}' = (G', T', A', \mathcal{P}', \mathcal{Q}')$ of the same order in polynomial time such that $\mathcal{W}'$ admits a definable root and $|A'| \geq |A|/3$.*

B. Trees admitting a definable ordering

In this section we show how to prune a given rooted tree T with maximum degree 3, so that there is an MSO_2-formula (not depending on T) which at each branching node of the tree distinguishes between the left and the right sub-tree. Assume we are given a sub-cubic tree T with a root r and a set X of vertices of the tree marked as *good* and we would like to retain as many good vertices as possible. Throughout this section, X will always denote the set of good vertices; and we assume $\deg_T(r) = 1$. We use the following notation:

- If $v \in V(T)$ then the children of v are all neighbours of v not on the unique path from v to r.
- A *leaf* of T is a node of degree 1 in T, except r.
- A vertex is called *leafy* if it has a leaf as a neighbour.
- A *branching vertex* of T is a node of degree 3 in T.
- A *proper branching vertex* is a branching vertex that is not leafy.
- We call a leaf a *good leaf*, if it is a good vertex.
- An *artificial leaf* is a leaf of T that is not a good leaf.

- Let $v \in V(T)$ be a vertex with child $u \in V(T)$ and $e = \{v, u\}$. The sub-tree T_u of T *rooted at* u is the component of $T - e$ containing u. The *extended sub-tree* of u is defined as $T_u \cup e$.
- $\text{SUBTREE}_i(v)$ denotes the extended sub-tree of the i^{th} child of v, where we number the children arbitrarily.
- $\text{NBV}(T, v)$: next (i.e. closest) branching vertex to v in T_v; or the leaf of T_v if T_v is a path; is defined only if v has degree 1 in T_v.
- $g_X(T) := |X \cap V(T)|$ is the number of good vertices in T. We omit the index $\cdot_X$ if it is clear from context.

Definition 4.10. *Let (T, r) be a rooted sub-cubic tree;*

- *two vertices $u, v \in V(T)$ are* topological neighbours *if they are linked by a path whose inner vertices all have degree 2 in T;*
- *a branching vertex v is called* properly marked *if it has a leaf or a leafy vertex as a topological neighbour in the sub-tree rooted at v; and*
- *T is called* properly marked *if every branching vertex of T is properly marked.*

We now define a pruning algorithm $\text{PRUNE}(T, r)$ which, given a rooted sub-cubic tree (T, r) outputs a tree (T', r) that is properly marked.

Algorithm $\text{PRUNE}(\boldsymbol{T}, \boldsymbol{r})$**.**

Input. sub-cubic rooted tree (T, r) with $\deg_T(r) = 1$.

Output. a properly marked sub-cubic rooted tree (T', r).

If T is a simple path than return T. Otherwise, let $v := \text{NBV}(T, r)$, R be the path from r to v, $T_1 := \text{SUBTREE}_1(v)$, and $T_2 := \text{SUBTREE}_2(v)$ with $g(T_1) \leq g(T_2)$.

1) If one of T_1, T_2 is a path, say T_i, return the tree obtained from T by replacing T_{3-i} by $\text{PRUNE}(T_{3-i})$.
2) Otherwise, let $u_1 := \text{NBV}(T_1, v)$. Let $T_{11} := \text{SUBTREE}_1(T_1, u_1)$ and $T_{12} := \text{SUBTREE}_2(T_1, u_1)$ with $g(T_{11}) \leq g(T_{12})$. Let T_1' be the tree obtained from T_1 by cutting T_{11} down to a single edge and replacing T_{12} by $T_{12}' := \text{PRUNE}(T_{12}, u_1)$. Finally, return T' as the union of R, T_1', and $T_2' := \text{PRUNE}(T_2)$.

Lemma 4.11. *Let (T, r) be a rooted sub-cubic tree and $X \subseteq V(T)$. T contains a properly marked sub-tree T' such that $g_X(T') \geq g_X(T)^{\frac{2}{3}}$. Furthermore, T' can be computed in polynomial time on input (T, r).*

Proof. Let $T' := \text{PRUNE}(T, r)$. We claim that (T', r) fulfills the requirements of the lemma. We prove the claim by induction on the order $n := |T|$ of T. If T is a path there is nothing to show. Otherwise, the fact that T' is properly marked is immediate from our recursive construction by induction. It remains to bound the number of good vertices that remain after the pruning. We first observe that for all $\beta \geq \frac{1}{2}$

$$\left(\frac{1-\beta}{2}\right)^{\frac{2}{3}} + \beta^{\frac{2}{3}} \geq 1\,. \tag{1}$$

If q, q_1, q_2 are non-negative integers with $q = q_1 + q_2$ and $q_1 \leq q_2$, we have $q_2 = \beta q$ and $q_1 = (1-\beta)q$, for some $\beta \geq \frac{1}{2}$. Hence, we obtain with Inequality (1)

$$\begin{aligned} q_1^{\frac{2}{3}} + q_2 \geq q_1 + q_2^{\frac{2}{3}} \geq q_1^{\frac{2}{3}} + q_2^{\frac{2}{3}} &\geq (\frac{q_1}{2})^{\frac{2}{3}} + q_2^{\frac{2}{3}} \\ &= q^{\frac{2}{3}} \cdot ((\frac{1-\beta}{2})^{\frac{2}{3}} + \beta^{\frac{2}{3}}) \\ &\geq q^{\frac{2}{3}} = (q_1 + q_2)^{\frac{2}{3}}\,. \end{aligned} \tag{2}$$

Let v, R. T_1, and T_2 be defined as in the algorithm. Define $q_0 := g(R - v)$, $q_1 := g(T_1 - v)$, $q_2 := g(T_2)$, $q := q_1 + q_2$, and $q' := g(T_v')$. First, note that if suffices to show $q' \geq q^{\frac{2}{3}}$ since this implies

$$g(T') = q_0 + q' \geq q_0 + q^{\frac{2}{3}} \geq (q_0 + q)^{\frac{2}{3}} = g(T)^{\frac{2}{3}}$$

by Inequality (2). Consider the following cases:

(i) If T_1 is a path, then $q' \geq q_1 + q_2^{\frac{2}{3}} \geq q^{\frac{2}{3}}$ by Inequality (2). Similarly, if T_2 is a path, then $q' \geq q_1^{\frac{2}{3}} + q_2 \geq q^{\frac{2}{3}}$.

(ii) Otherwise, let T_{12}' and T_2' be defined as in Step 2 of the algorithm and let $q_2' := g(T_2')$ and $q_{12}' := g(T_{12}')$. Furthermore, let P be the path from u to v excluding u and v and let $q_P := g(P)$. Using Inequality (2) twice more, we obtain

$$\begin{aligned} q' = q_P + q_{12}' + q_2' &\geq q_P + (\frac{q_1 - q_P}{2})^{\frac{2}{3}} + q_2^{\frac{2}{3}} \\ &\geq (\frac{q_1}{2})^{\frac{2}{3}} + q_2^{\frac{2}{3}} \geq q^{\frac{2}{3}}\,. \end{aligned}$$

□

Once we have a properly marked tree, it is possible to mark left and right sub-trees in a proper way using parity considerations, as follows.

Definition 4.12. *Let (T, r) be a sub-cubic tree rooted at vertex r of degree 1 and X a set of good vertices that lies flat in T. We say the tuple (T, r, X)* admits a definable order *if for all branching vertices v with extended sub-trees $T_i := \text{SUBTREE}_i(v)$, for $i = 1, 2$, exactly one of the following is true. Along with the following conditions we will label some sub-trees as* left *and others as* right*.*

1) *At least one of T_1, T_2 is a single edge, say T_1. If T_2 is also a single edge, then T_1 and T_2 are incomparable; otherwise T_1 is left and T_2 is right.*
2) *Exactly one of T_1, T_2 is a simple path, say T_1. Then T_1 is left and T_2 is right.*
3) *Let u_i be the next proper branching vertex of T_i if one exists; otherwise let u_i be the leaf of T_i that is farthest away from v. Let P_i be the path connecting v and u_i in T_i. We define g_i to be the number of vertices on P_i that are good or leafy. We require that exactly one of g_1, g_2 is odd, say g_1; then T_1 is left and T_2 is right.*

The canonical order $\leq_T$ *of (T, r) is defined as follows. Let $x \neq y \in V(T)$ and let v be the common ancestor of x, y.*

Then $x \leq_T y$ if and only if x is in the left sub-tree of v and y in the right.

Lemma 4.13. *Let (T, r) be a sub-cubic tree rooted at vertex r of degree 1 and $X \subseteq V(T)$ a given set of good vertices that lies flat in T. T contains a sub-tree T' and a set $X' \subseteq X \cap T'$ with $|X'| \geq |X|^{\frac{2}{3}}/2$ such that (T', r, X') admits a definable order and X' is* totally *ordered by the canonical order $\leq_{T'}$. Furthermore, T' can be computed in polynomial time.*

Finally, we can relate our construction to MSO_2 by the following lemma; its proof is immediate from Definition 4.12:

Lemma 4.14. *Let (T, r) be a sub-cubic tree and $X \subseteq V(T)$ such that (T, r, X) admits a definable ordering. Let* MSO_2*-formulas $\varphi_R(v)$ and $\varphi_X(v)$ defining the root of T and the set X, respectively, be given. Then there is an* MSO_2*-formula $\varphi_{\preceq}(x, y)$ which defines the canonical order $\leq_T$.*

C. Tree-Ordered Webs

In this section we show how to prune the tree T of a given nice full tree-web $(G, T, A, \mathcal{P}, \mathcal{Q})$, so that there is an MSO_2-formula which can detect the nodes of T in G and at each branching node of the tree distinguishes between the left and right sub-tree. We use the same notion of a *topological neighbour* as in the previous section, only that now we consider degrees in G, not in T; this does make an important difference, since good vertices have degree 2 in T but degree 3 in G. Additionally, we call a vertex $v \in V(G)$ *special in G* if it has degree 4 or has degree 3 and is *not* properly marked, i.e. does not have a leaf or a leafy vertex as a topological neighbour. We denote by *spec*(G) the set of special vertices in G.

Definition 4.15. *A* tree-ordered web *of order ℓ is a tuple $(G, T, r, A, \mathcal{P}, \mathcal{Q})$ such that*

1) *$(G, T, A, \mathcal{P}, \mathcal{Q})$ is a tree-web of order ℓ admitting the definable root r,*
2) *A is the set of vertices of degree 3 in G not in spec(G) but having a topological neighbour in spec(G),*
3) *T is contained in a component of $G -$ spec(G),*
4) *(T, r, A) admits a definable order.*

Lemma 4.16. *There exists a constant c such that if $\mathcal{W}_0 = (G_0, T_0, A_0, \mathcal{P}_0, \mathcal{Q}_0)$ is a given nice full tree-web of order $c\ell$, then there exists a tree-web $\mathcal{W} = (G, T, A, \mathcal{P}, \mathcal{Q})$ with $\mathcal{W} \subseteq \mathcal{W}_0$ and a vertex $r \in V(G)$ such that $(G, T, r, A, \mathcal{P}, \mathcal{Q})$ is a tree-ordered web of order ℓ with $|A| \geq 15\ell$; furthermore, $\mathcal{W}$ can be computed in polynomial time.*

A main ingredient of the proof of Lemma 4.16 is the observation that Definition 4.12 allows us to cut a good path and make it an artificial leaf without destroying the definable order. Hence, we can cut away about every second good path to ensure that vertices of the tree do not land in *spec*(G).

Figure 3. A (a) single and a (b) double cross.

We also observe that if the number of the leaves of the tree is large enough, we can just keep the good paths starting at the leaves; and otherwise, the number of proper branching vertices is small and we do not need to cut away too many good paths.

The proof of the following lemma is immediate from Definition 4.15, Lemma 4.14, and the fact that the special vertices *spec*(G) can easily be defined:

Lemma 4.17. *Given a tree-ordered web $\mathcal{W} = (G, T, r, A, \mathcal{P}, \mathcal{Q})$, there exist* MSO_2*-formulas $\varphi_T(x)$, $\varphi_R(x)$, $\varphi_A(x)$, $\varphi_{PQ}(x)$, and $\varphi_{\preceq}(x, y)$, defining the tree T, its root r, the set of good vertices A, the vertices of the grid-like minor, i.e. $V(\bigcup \mathcal{P} \cup \bigcup \mathcal{Q})$, and the canonical order $\leq_T$, respectively.*

D. Labelling Tree-Ordered Webs

We will show next how to encode a word $w := w_1 \ldots w_t \in \{0, 1\}^\star$ in a tree-ordered web of order $2t$. We first need the following simple combinatorial lemma.

Lemma 4.18. *Let G be a directed graph on k vertices with maximum outdegree d. Then G contains an independent set of size $\left\lceil \frac{k}{2d+1} \right\rceil$ which can be computed in polynomial time.*

A *single cross* is a sub-cubic tree with four leaves having the shape depicted in Fig. 3 (a); a *double cross* is a sub-cubic tree with five leaves having the shape depicted in Fig. 3 (b) (where the dashed lines indicate paths). The rightmost vertex of each cross, as drawn in Fig. 3, is called the *base* of the cross.

Definition 4.19. *A* labelled tree-ordered web of order ℓ and length k *is a tuple $\mathcal{W} := (G, T, r, A, \mathcal{P}, \mathcal{Q}, X, C)$ where*

1) *$((G - C) \cup X, T, r, A, \mathcal{P}, \mathcal{Q})$ is a tree-ordered web of order ℓ,*
2) *the root r does not have a leafy vertex as a topological neighbour,*
3) *C is a set of disjoint single and double crosses,*
4) *$X = T \cap C$ is the set of bases of the crosses in C and lies flat in T,*
5) *$|C| = |X| = |A| = k$,*
6) *if $X = \{x_1, \ldots, x_k\}$ and $A = \{v_1, \ldots, v_k\}$ then $x_1 \leq_T v_1 \leq_T x_2 \leq_T \cdots \leq_T x_k \leq_T v_k$.*

The word encoded by *$\mathcal{W}$ is $w := w_1 \ldots w_k \in \{0, 1\}^k$ with $w_i := 0$ if x_i is the base of a single cross in C and $w_i := 1$ if x_i is the base of a double cross of C. $\mathcal{W}$ is called* configurable *if C consists only of double crosses.*

A labelled tree-ordered web encoding the word 010 is indicated in Figure 2.

Lemma 4.20. *For $\ell \geq 3$, let $\mathcal{W} = (G, T, r, A, \mathcal{P}, \mathcal{Q})$ be a given tree-ordered web of order 2ℓ with $|A| \geq 30\ell$. There exists a configurable labelled tree-ordered web $\mathcal{W}' = (G', T', r', A', \mathcal{P}', \mathcal{Q}', X', C')$ of order ℓ and length ℓ with $G' \subseteq G$ that can be computed in polynomial time.*

Proof sketch. First, note that any good path $P \in \mathcal{P}$ can be easily transformed to a double cross by destroying at most 7 good paths. Consider a digraph $\mathcal{D}$ having a vertex u_P for each good path in $\mathcal{P}$ and a directed edge from u_P to $u_{P'}$ if turning P into a double cross destroys P'. The maximum out-degree of this digraph is 7 and hence, by Lemma 4.18, there exists a set $Y_0 \subseteq A$ of size at least $\frac{|A|}{15} \geq 2\ell$ of good vertices, so that the vertices in $\mathcal{D}$ that correspond to the good paths starting at Y_0 build an independent set in $\mathcal{D}$.

Let $Y := \{y_1, \ldots, y_{2\ell}\}$ be a subset of exactly 2ℓ vertices of Y_0 such that $y_1 \leq_T y_2 \leq_T \cdots \leq_T y_{2\ell}$. We define $X' := \{y_i \,|\, i \text{ is odd}\}$ and $A' := \{y_i \,|\, i \text{ is even}\}$. We transform every good path starting at a vertex in X' into a double cross and remove all the paths that get destroyed. Let C' be the set of double crosses we obtain this way. The rest of the proof follows easily. □

Lemma 4.21. *Given a labelled tree-ordered web $\mathcal{W} = (G, T, r, A, \mathcal{P}, \mathcal{Q}, X, C)$, there exist* MSO$_2$*-formulas $\varphi_T(x)$, $\varphi_R(x)$, $\varphi_A(x)$, $\varphi_X(x)$, $\varphi_C(x)$, $\varphi_{PQ}(x)$, and $\varphi_{\preceq}(x, y)$ defining T, r, A, X, C, $\mathcal{P} \cup \mathcal{Q}$, and the canonical order $\leq_T$, respectively.*

Proof. Since r does not have a leafy vertex as a topological neighbour, the crosses are distinguishable from r. Hence, we can easily obtain an MSO$_2$-formula $\varphi_C(x)$ to identify the vertices in the crosses and then apply Lemma 4.17 to obtain our result. □

The definition below is needed in Section V:

Definition 4.22. *If $\mathcal{W}$ is a labelled tree-ordered web of order ℓ^d and length ℓ encoding a word $w = w_1 \ldots w_\ell$, we say that $\mathcal{W}$* encodes *w* with power *d.*

Theorem 4.23 sums up the algorithmic part of this work:

Theorem 4.23. *Let a word $w = w_1 \ldots w_\ell \in \{0, 1\}^\star$, a graph G, and an integer d be given. There is a constant c, so that if the tree-width of G is at least $c\ell^{14d}$ then G contains either an $\ell^d \times \ell^d$-wall or a labelled tree-ordered web $\mathcal{W}$ that encodes w with power d. Furthermore, either outcome can be computed in polynomial time.*

V. Parameterized Intractability of MSO$_2$ Model Checking

In this section we prove Theorem 1.2. We first show the intractability of MSO$_2$ on walls and then lift this to show the general result. For this, we first recall the well-known fact that MSO$_2$ is intractable on coloured walls. Recall that from the results of the previous section, given a word w and a graph G of large enough tree-width, we construct either a wall encoding w or a labelled tree-ordered web encoding w. For either outcome we will define an MSO$_2$-interpretation of coloured walls in these structures which will allow us to transfer the intractability results from coloured walls to these structures and hence to the general case of Theorem 1.2.

A. MSO$_2$-*Interpretations*

We first recall briefly the concepts of interpretations (see e.g. [5]).

Definition 5.1. *Let σ and τ be signatures and let $\overline{X}$ be a tuple of monadic second-order variables. An* interpretation *of τ in σ with parameters $\overline{X}$ is a tuple $\Theta := \big(\varphi_{valid}, \varphi_{univ}(x), \varphi_{\sim}(x, y), (\varphi_R(\overline{x}))_{R \in \tau}\big)$ of* MSO$_2[\sigma \dot\cup \overline{X}]$*-formulas, where the arity of $\overline{x}$ in $\varphi_R(\overline{x})$ is $ar(R)$, such that for all σ-structures A and assignments $\overline{Y} \subseteq U(A)$ to $\overline{X}$ with $(A, \overline{Y}) \models \varphi_{valid}$, $\varphi_{\sim}$ defines an equivalence relation on $\varphi_{univ}(A)$.*

For an interpretation Θ we will denote φ_{valid} by $\varphi_{valid}(\Theta)$. With any interpretation Θ we associate a map taking a σ-structure A and $\overline{Y} \subseteq U(A)$ such that $(A, \overline{Y}) \models \varphi_{valid}$ to a τ-structure H with universe $U(H) := \varphi_{univ}(A, \overline{Y})_{|\varphi_{\sim}(A, \overline{Y})} := \{[v]_\sim : (A, \overline{Y}) \models \varphi_{univ}(v)\}$ where $[v]_\sim$ denotes the equivalence class of v under $\varphi_\sim(A, \overline{Y})$. For $R \in \tau$ of arity $r := ar(R)$ we define $R(H) := \{([a_1], \ldots, [a_r]) : (A, \overline{Y}) \models \varphi_R(a_1, \ldots, a_r)\}$.

Furthermore, any interpretation Θ also defines a translation of MSO$_2[\tau]$-formulas φ to MSO$_2[\sigma]$-formulas $\Theta(\varphi)$ by replacing occurrences of relations $R \in \tau$ by their defining formulas $\varphi_R \in \Theta$ in the usual way (see [5] for details) so that the following lemma holds.

From now on we will always let $\sigma = \tau := \sigma_{graph}$ and therefore speak about interpretations without any reference to specific signatures.

Lemma 5.2 (Interpretation Lemma)**.** *Let Θ be an* MSO$_2$*-interpretation with parameters $\overline{X}$. For any σ_{graph}-structure A and assignment $\overline{Y} \subseteq U(A)$ to $\overline{X}$ s.t. $(A, \overline{Y}) \models \varphi_{valid}(\Theta)$, and any* MSO$_2$*-sentence φ we have $\Theta(A, \overline{Y}) \models \varphi$ if, and only if, $(A, \overline{Y}) \models \Theta(\varphi)$.*

B. MSO$_2$ *on Coloured Elementary Walls*

Definition 5.3. *The signature σ_{wall} of coloured walls is defined as $\sigma_{wall} := \{V, E, \in, C_0, C_1\}$, where V, E, C_0, C_1 are unary relation symbols and $\in$ is a binary relation symbol. A σ_{wall}-structure W is a* coloured elementary *$l \times l$*-wall *if its σ_{graph}-reduct $W_{|\{V,E,\in\}}$ is an elementary $l \times l$-wall according to Definition 2.3. W encodes a word $w := w_1 \ldots w_n \in \Sigma^n$ with power d if $l \geq n^d$ and if $\{v_{1,i} : 1 \leq i \leq l\}$ are the vertices on the bottom row then $v_{1,i} \in C_0$ if $w_i = 0$ and $v_{1,i} \in C_1$ if $w_i = 1$, for all $1 \leq i \leq n$, and $C_0 \cap C_1 = \varnothing$.*

The following theorem is part of the folklore.

Theorem 5.4. *For $d \geq 2$ let $\mathfrak{W}_d$ be the class of coloured elementary walls encoding words with power d. Then $\mathrm{MC}(\mathrm{MSO}_2, \mathfrak{W}_d)$ is not in XP unless $P = \mathrm{NP}$.*

The theorem follows immediately from the following lemma, whose proof is standard.

Lemma 5.5. *Let M be a non-deterministic n^d-time bounded Turing-machine. There is a formula $\varphi_M \in \mathrm{MSO}_2$ such that for all words $w \in \Sigma^\star$, if W is a coloured elementary wall encoding w with power d, then $W \models \varphi_M$ if, and only if, M accepts w. Furthermore, the formula φ_M can be constructed effectively from M. The same holds if M is an alternating Turing-machine with a bounded number of alternations, as they are used to define the polynomial-time hierarchy.*

C. MSO_2 on Uncoloured Walls

The previous paragraph stated the intractability of MSO_2 on coloured elementary walls. As one possible outcome of Section IV we get an uncoloured wall W, not necessarily elementary, of sufficient size. In the absence of colours we will encode a word w in W by taking a suitable sub-graph $W' \subseteq W$ as follows.

Let $w := w_1, \dots, w_n \in \{0,1\}^*$ be a word of length n, let $d \geq 1$ and let $m := n^d + 2$. The aim is to encode w in a wall W of order at least $m \times m$. Let $v_0, \dots, v_{m-1}$ be the nails (see 2.3) on the bottom-row $R \subseteq W$ of W and, for $0 \leq i < m-1$, let $P_i \subseteq R$ be the sub-path connecting v_i and v_{i+1}. Let $W' \subseteq W$ be the wall obtained from W by deleting the internal vertices of P_0 and for each $1 \leq i \leq n$ such that $w_i = 0$ the internal vertices of P_i. All other paths remain unchanged. We say that W' *encodes* w *with power* d. The reason we delete P_0 but not P_{m-1} is that this defines a unique ordering on the bottom-row, the left-most path is deleted but the right-most is not, needed to read the word w in the correct order. The following is now easily seen.

Theorem 5.6. *There is an MSO_2-interpretation Θ from σ_{wall} in σ_{graph} such that if W is an uncoloured wall encoding $w \in \Sigma^*$ with power d then $\Theta(W)$ is a coloured elementary wall encoding w with power d.*

D. Defining Labelled Tree-Ordered Webs

The aim of this section is to show that we can define a coloured elementary wall encoding a word w in a labelled tree-ordered web encoding w. The main result of this part is the following theorem.

Theorem 5.7. *There is an MSO_2-interpretation Θ such that if $(G, T, r, A, \mathcal{P}, \mathcal{Q}, X, C)$ is a labelled tree-ordered web encoding a word w with power d, then $\Theta(G)$ is a coloured elementary wall encoding w with power d.*

Proof. We will define the interpretation in a sequence of steps and will illustrate the formulas by a labelled tree-ordered web $\mathcal{W} = (G, T, r, A, \mathcal{P}, \mathcal{Q}, X, C)$ encoding a word w of length ℓ with power d. The actual formulas will not depend on $\mathcal{W}$ in any form.

By Lemma 4.21, there exist MSO_2-formulas $\varphi_T(x)$, $\varphi_R(x)$, $\varphi_A(x)$, $\varphi_X(x)$, $\varphi_C(x)$, $\varphi_{PQ}(x)$, and $\varphi_\preceq(x,y)$ defining T, r, A, X, C, $\mathcal{P} \cup \mathcal{Q}$, and the canonical order $\leq_T$, respectively. Essentially, we now have formulas which, on G as above, define the labelled tree-ordered web $\mathcal{W}$.

What is left to do is to define formulas which generate a wall from the grid-like minor $(\mathcal{P}, \mathcal{Q})$ so that the bottom-row of the wall is connected to the vertices in $A = \{v_1, \dots, v_\ell\}$ in the correct order, where ℓ is the length of the word w. This is by far the most complex part of the interpretation. But fortunately we can adapt formulas defined in [8] for this purpose. The actual formulas are very long and tedious. We therefore refrain from repeating them here and refer the full version of [8] at http://arxiv.org/abs/0904.1302.

Let $k := \ell^d$; by Definition 4.19 we know that $\mathcal{I}(\mathcal{P}, \mathcal{Q})$ contains a sub-division of K_{k^2}, so that ℓ of its nails are paths from $\mathcal{P}$ that start at the vertices $v_1 \leq_T \cdots \leq_T v_\ell$ of A. We can consider a subgraph of this K_{k^2} sub-division to obtain a $k \times k$-wall in $\mathcal{I}(\mathcal{P}, \mathcal{Q})$, so that the first ℓ nails of the bottom row of the wall are adjacent to A. Suitably adapting the formulas defined in [8], we obtain formulas $\varphi_{valid}, \varphi_{univ}, \varphi_\sim, \varphi_V, \varphi_E$, and $\varphi_\in$ needed for the interpretation Θ. What is left to do is to define the colouring. However, it is easily seen that there are MSO_2-formulas $\varphi'_{C_0}(x), \varphi'_{C_1}(x)$, using $\varphi_C, \varphi_X, \varphi_A$ and $\varphi_\preceq$ from Lemma 4.21, so that for $X := \{x_1, \dots, x_\ell\}$, with $x_1 \leq_T v_1 \leq_T x_2 \cdots \leq_T x_\ell \leq v_\ell$, $v_i \in \varphi'_{C_0}(G)$ if x_i is adjacent to a single cross and $v_i \in \varphi'_{C_1}(G)$ if x_i is adjacent to a double cross in C. From this, formulas $\varphi_{C_0}(x, P, Q, A)$ and $\varphi_{C_1}(x, P, Q, A)$ defining the correct colouring of the wall W' can be easily defined. This concludes the proof of the theorem. □

E. Proof of Theorem 1.2.

In this section we complete the proof of Theorem 1.2. We first prove Part 1.

Suppose, $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \in \mathrm{XP}$, i.e. there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that given $G \in \mathcal{C}$ and $\varphi \in \mathrm{MSO}_2$ we can decide $G \models \varphi$ in time $\mathcal{O}(|G|^{f(|\varphi|)})$. Let M be a non-deterministic Turing-machine deciding SAT in quadratic time and let φ_M be the formula constructible from M as defined in Lemma 5.5.

Let Θ_1 be the interpretation from the Theorem 5.6 and let Θ_2 be the interpretation from Theorem 5.7. Define $\varphi_M^1 := \Theta_1(\varphi_M)$ and $\varphi_M^2 := \Theta_2(\varphi_M)$.

Let $w \in \{0,1\}^*$ be a word of which we want to decide whether $w \in$ SAT, let $\ell := |w|$ and $t := 2c\ell^{28}$, where c is the constant from Theorem 4.23. As the treewidth of $\mathcal{C}$ is strongly unbounded by $\log^{28\gamma} n$, there are $\varepsilon < 1$ and a polynomial $p(n)$ of degree less than γ such that $\mathcal{C}$ contains a graph G with $\mathrm{tw}(G) \geq \log^{28\gamma} |G|$ and $t \leq \mathrm{tw}(G) \leq$

$p(t)$ and G can be computed in time $2^{|w|^{\varepsilon}}$; note that this implies that $|G| \leq 2^{p(2c\ell^{28})^{\frac{1}{28\gamma}}} \leq 2^{|w|^{\delta}}$, for some $\delta < 1$. By Theorem 4.23, G either contains a) a wall W_w encoding w with power 2 as a subgraph or b) a labelled tree-ordered web $\mathcal{W} = (H, T, r, A, \mathcal{P}, \mathcal{Q}, X, C)$ encoding w with power 2. Note that we need an encoding with power 2 because M needs $|w|^2$ space cells and computation steps to decide $w \in$ SAT.

In case $a)$, as $\mathcal{C}$ is closed under sub-graphs, $W_w \in \mathcal{C}$ and we can therefore decide $W_w \models \varphi_M^1$ in time $|W_w|^{f(|\varphi_M^1|)} \leq |G|^{f(|\varphi_M^1|)} \leq (2^{|w|^{\delta}})^{f(|\varphi_M^1|)} = 2^{f(|\varphi_M^1|)|w|^{\delta}} = 2^{o(|w|)}$. By construction, $W_w \models \varphi_M$ if, and only if, M accepts w if, and only if, $w \in$ SAT.

In case b), $H \in \mathcal{C}$ as H is a subgraph of G. We can therefore decide $H \models \varphi_M^2$ in time $|H|^{f(|\varphi_M^2|)} \leq |G|^{f(|\varphi_M^2|)} \leq (2^{|w|^{\delta}})^{f(|\varphi_M^2|)} = 2^{f(|\varphi_M^2|)|w|^{\delta}} = 2^{o(|w|)}$. By construction, $H \models \varphi_M$ if, and only if, M accepts w if, and only if, $w \in$ SAT.

Hence, in both cases a) and b) we can decide $w \in$ SAT in time $2^{o(|w|)}$. This shows Part 1.

To show Part 2, we use the same proof idea. Let P be a language in the polynomial-time hierarchy and let M be an alternating Turing-machine with bounded alternation deciding P in time n^k. We use essentially the same proof as above but, given a word w, we construct a graph G which contains a wall or a labelled tree-ordered web encoding w with power k. The rest follows then as before.

This concludes the proof of Theorem 1.2. It is easily seen that the proof can be adapted to classes of graphs closed under spanning sub-trees, i.e. edge deletion: instead of taking sub-graphs we simply delete all edges no longer needed and make the MSO_2-formulas ignore isolated vertices.

Corollary 5.8. *If $\mathcal{C}$ is closed under spanning sub-graphs and the tree-width of $\mathcal{C}$ is strongly unbounded by $\log^{28} n$, then $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \notin \mathrm{XP}$ unless SAT can be solved in sub-exponential time. If the tree-width of $\mathcal{C}$ is not poly-logarithmically bounded then $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \notin \mathrm{XP}$ unless all problems in the polynomial-time hierarchy can be solved in sub-exponential time.*

VI. Conclusion

We have presented a strong intractability result for MSO_2 on graph classes of unbounded tree-width. In comparison to Courcelle's theorem, Courcelle's theorem requires the tree-width to be constant whereas our result refers to classes whose tree-width is essentially not bounded logarithmically. However, it seems difficult to close this gap as we believe that there are classes of graphs whose tree-width is only bounded by $\log^{1-\varepsilon} n$ but which are closed under sub-graphs and satisfy the other conditions above, but do admit tractable MSO_2-evaluation. On the other hand, this is very unlikely to be the case for all classes of logarithmic tree-width. Exploring tractability and intractability of MSO_2 on classes of unbounded tree-width, but bounded by $\log n$, might lead to interesting new results on the boundary of MSO_2-tractability.

The results reported in this paper refer to MSO_2, i.e. MSO with quantification over sets of edges. For MSO without edge set quantification, referred to as MSO_1, it can be shown that MSO_1 is tractable on any class $\mathcal{C}$ of graphs of bounded *clique width*. Again not much is known about MSO_1 and graph classes of unbounded clique-width and it would be very interesting to establish similar results as in this paper for the case of clique-width. This, however, is much more difficult as there is no good obstruction similar to grid-like minors.

References

[1] B. Courcelle, "Graph rewriting: An algebraic and logic approach," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier, 1990, vol. 2, pp. 194 – 242.

[2] R. Diestel, *Graph Theory*. Springer, 2005.

[3] J. Flum and M. Grohe, *Parameterized Complexity Theory*. Springer, 2006, iSBN 3-54-029952-1.

[4] M. Grohe, "Logic, graphs, and algorithms," in *Logic and Automata – History and Perspectives*, T. J.Flum, E.Grädel, Ed. Amsterdam University Press, 2007.

[5] W. Hodges, *A shorter model theory*. Cambridge University Press, 1997.

[6] S. Kreutzer and S. Tazari, "On brambles, grid-like minors, and parameterized intractability of monadic second-order logic," in *Symposium on Discrete Algorithms (SODA)*, 2010.

[7] S. Kreutzer, "Algorithmic meta-theorems," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. TR09-147, 2009.

[8] ——, "On the parameterised intractability of monadic second-order logic," in *Proc. of Computer Science Logic (CSL)*, 2009.

[9] L. Libkin, *Elements of Finite Model Theory*. Springer, 2004.

[10] J. A. Makowsky and J. Mariño, "Tree-width and the monadic quantifier hierarchy," *Theor. Comput. Sci.*, vol. 1, no. 303, pp. 157–170, 2003.

[11] B. Reed and D. R. Wood, "Polynomial treewidth forces a large grid-like minor," 2008, arXiv:0809.0724v3 [math.CO].

[12] N. Robertson, P. Seymour, and R. Thomas, "Quickly excluding a planar graph," *J. Comb. Theory Ser. B*, vol. 62, no. 2, pp. 323–348, 1994.

[13] M. Vardi, "On the complexity of relational query languages," in *Proc. of the 14th Symposium on Theory of Computing (STOC)*, 1982, pp. 137–146.