The complexity of first-order and monadic second-order logic revisited

Markus Frick Martin Grohe

May 2, 2002

Abstract

The model-checking problem for a logic L on a class C of structures asks whether a given L-sentence holds in a given structure in C. In this paper, we give super-exponential lower bounds for fixed-parameter tractable model-checking problems for first-order and monadic second-order logic.

We show that unless PTIME = NP, the model-checking problem for monadic second-order logic on finite words is not solvable in time $f(k) \cdot p(n)$, for any *elementary* function f and any polynomial p. Here k denotes the size of the input sentence and n the size of the input word. We prove the same result for first-order logic under a stronger complexity theoretic assumption from parameterized complexity theory.

Furthermore, we prove that the model-checking problems for first-order logic on structures of degree 2 and of bounded degree $d \geq 3$ are not solvable in time $2^{2^{o(k)}} \cdot p(n)$ (for degree 2) and $2^{2^{2^{o(k)}}} \cdot p(n)$ (for degree d), for any polynomial p, again under an assumption from parameterized complexity theory. We match these lower bounds by corresponding upper bounds.

1. Introduction

Model-checking problems. We study the complexity of a fundamental algorithmic problem, the so-called *model-checking* problem: Given a sentence φ of some logic L and a structure $\mathcal{A}$, decide whether φ holds in $\mathcal{A}$. Model-checking and closely related algorithmic problems occur frequently in computer science, for example, in database theory, artificial intelligence, and automated verification. In this paper, we prove new lower bounds on the complexity of the model-checking problems for first-order and monadic second-order logic.

It is known that model-checking for both first-order and monadic second-order logic is PSPACE-complete [18, 20] and thus most likely not solvable in polynomial time. While this result shows that the problems are intractable *in general*, it does not say too much about their complexity in practical situations. Typically, we have to check whether a relatively *small* sentence holds in a *large* structure. For example, when evaluating a database query, we usually have a small query and a large database. Similarly, when verifying that a finite state system satisfies some property, the specification of the property in a suitable logic will usually be small compared to the huge state space of the system. When analysing the complexity of the problem, we should take this imbalance between the size of the input sentence and the size of the input structure into account.

Parameterized complexity theory. Parameterized complexity theory (see [7]) is a relatively new branch of complexity theory that provides the framework for a refined complexity analysis of problems whose instances consist of different parts that typically have different sizes. In this framework, a *parameterized problem* is a problem whose instances consist of two parts of sizes n and k, respectively. k is called the *parameter*, and the assumption is that k is usually small, small enough that an algorithm that is exponential in k may still be feasible. A parameterized problem is called *fixed-parameter tractable* if it can be solved in time $f(k) \cdot p(n)$ for an arbitrary computable function f and some polynomial p. The motivation for this

Authors' address: Division of Informatics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK.
Email: {mfrick,grohe}@dcs.ed.ac.uk

definition is that, since k is assumed to be small, the feasibility of an algorithm for the problem mainly depends on its behaviour in terms of n. Under this definition, a running time of $O(2^k \cdot n)$ is considered tractable, but running times of $O(n^k)$ or $O(k \cdot 2^n)$ are not, which seems reasonable.

A standard example of a fixed-parameter tractable problem is the parameterized vertex cover problem: Decide if a graph of size n has a vertex cover of size k. There is a simple $O(2^k \cdot n)$-algorithm for this problem, which means that finding small vertex covers, say of size 10, is possible in quite large graphs. On the other hand, no comparable algorithm is known for the parameterized clique problem. (Decide if a graph of size n has a clique of size k.) The best known algorithm for the clique problem has a running time of $n^{\Omega(k)}$. Indeed, it can be proved that the clique problem is complete for the parameterized complexity class W[1] under suitable reductions; it is generally believed that this class strictly contains the class FPT of all fixed-parameter tractable problems. Note that both the vertex cover and the clique problem are NP-complete, so classical complexity theory does not detect this difference between the complexities of the two problems.

Let us conclude our short discussion of parameterized complexity theory by remarking that although fixed-parameter tractability has proven to be a valuable concept allowing fine distinctions on the borderline between tractability and intractability, it seems somewhat questionable to admit *all* computable functions f for the parameter dependence of a fixed-parameter tractable algorithm. If f is doubly exponential or worse, an $O(f(k) \cdot n)$-algorithm can hardly be considered tractable. The main contribution of this paper to parameterized complexity theory is to show that there are natural fixed-parameter tractable problems requiring parameter dependences f that are doubly exponential or even non-elementary.

The parameterized complexity of model-checking problems. Model-checking problems have a natural parameterization in which the size k of the input sentence is the parameter. We have argued above that k is usually small in the practical situations we are interested in, so a parameterized complexity analysis is appropriate. Unfortunately, it turns out that the model-checking problem for first-order logic is complete for the parameterized complexity class AW[$*$], which is assumed to strictly contain FPT. Thus probably model-checking for first-order logic is not fixed-parameter tractable. Of course this implies that model-checking for the stronger monadic-second order logic is also most-likely not fixed-parameter tractable. As a matter of fact, it follows immediately from the observation that there is a monadic second-order sentence saying that a graph is 3-colourable that model-checking for the stronger monadic-second is not fixed-parameter tractable unless P = NP.

It is interesting to compare these intractability results for first-order logic and monadic second-order logic with the following: The model-checking problem for linear time temporal logic LTL is solvable in time $2^{O(k)} \cdot n$ [16], making it fixed-parameter tractable and also tractable in practice. On the other hand, model-checking for LTL is PSPACE-complete (as it is for first-order and monadic second-order logic). So again parameterized complexity theory helps us establishing an important distinction between problems of the same classical complexity.[2] We may argue, however, that the comparison between LTL model-checking and first-order model-checking underlying these results is slightly unfair. As the name linear time temporal logic indicates, LTL only speaks about a linearly ordered sequence of events. On an arbitrary structure, an LTL formula can thus only speak about the paths through the structure. First-order formulas do not have such a restricted view. It is therefore more interesting to compare LTL and first-order logic on *words*, which are the natural structures describing linear sequences of events. A well-known result of Kamp [14] states that LTL and first-order logic have the same expressive power on words. And indeed, model-checking for first-order logic and even for monadic second-order logic is fixed-parameter tractable if the input structures are restricted to be words. This is a consequence of Büchi's theorem [2], saying that for every sentence of monadic second-order logic one can effectively find a finite automaton accepting exactly those words in which the sentence holds. A fixed-parameter tractable algorithm for monadic second-order model-checking on words may proceed as follows: It first translates the input sentence into an equivalent automaton and then tests in linear time whether this automaton accepts the input word. But note that since there is no elementary bound for the size of a finite automaton equivalent to a given first-order or monadic

[2] A critical reader may remark that this distinction between the complexities of LTL model-checking and first-order model-checking was known before anybody thought of parameterized complexity-theory. In some sense, this is true, but how can we be sure that there is no $2^{O(k)} \cdot n$ model-checking algorithm for first-order model-checking? The role of parameterized-complexity theory is to give evidence for this.

second-order sentence [19], the parameter dependence of this algorithm is non-elementary, thus it does not even come close to the $2^{O(k)} \cdot n$ model-checking algorithm for LTL. Of course this does not rule out the existence of other, better fixed-parameter tractable algorithms for first-order or monadic-second-order model-checking.

Our results. Our first theorem shows that there is no fundamentally better fixed-parameter tractable algorithm for first-order and monadic second-order model-checking on the class of words than the automata based one described in the previous paragraph.

Theorem 1. *(1) Assume that* PTIME $\neq$ NP. *Then there is no model-checking algorithm for monadic second-order logic on the class of words whose running time is bounded by $f(k) \cdot p(n)$ for an elementary function f and a polynomial p.*

(2) Assume that FPT $\neq$ W$[t]$ *for some $t \geq 1$. Then there is no model-checking algorithm for first-order logic on the class of words whose running time is bounded by $f(k) \cdot p(n)$ for an elementary function f and a polynomial p.*

Here k denotes the size of the input sentence of the model-checking problem and n the size of the input word.

Recall that FPT denotes the class of all fixed-parameter tractable problems. The classes W$[t]$, for $t \geq 1$, form a hierarchy of larger and larger parameterized complexity classes, which are widely believed to be different from FPT (see, for example, [7]). It is worth mentioning that FPT $=$ W$[1]$ would imply that 3SAT is solvable in deterministic time $2^{o(n)}$ [1, 7]. A function $f : \mathbb{N} \to \mathbb{N}$ is *elementary* if it can be formed from the successor function, addition, subtraction, and multiplication using concatenations, projections, bounded additions and bounded multiplications (of the form $\sum_{z \leq y} g(\bar{x}, z)$ and $\prod_{z \leq y} g(\bar{x}, z)$). The crucial fact for us is that a function f is bounded by an elementary function if, and only if, it is bounded by an h-fold exponential function for some fixed h (see, for example, [4]).

To prove the theorem, we use similar coding tricks as those that can be used to prove that there is no elementary algorithm for deciding the satisfiability of first-order sentences over words [19]. When we started to think about this problem, it was not clear at all to us which were the right complexity theoretic assumptions. We find it quite surprising that we ended up with fairly standard assumptions. It is not obvious what PTIME $\neq$ NP has to do with our non-elementary lower bound. And it is remarkable that the assumption FPT $\neq$ W$[t]$ for some $t \geq 1$, which is usually used to prove that problems are not fixed-parameter tractable, is used here to prove lower bounds for problems that are fixed-parameter tractable.

Model-checking for first-order and monadic second-order logic is known to be fixed-parameter tractable on several other classes of structures besides words: Model-checking for monadic second-order logic is also fixed-parameter tractable on trees and graphs of bounded tree-width [3]. The latter is a well-known theorem due to Courcelle [3] playing a prominent role in parameterized complexity theory. Theorem 1 implies that the parameter dependence of monadic-second-order model-checking on trees and and graphs of bounded tree-width is also non-elementary. In addition to trees and graphs of bounded tree-width, model-checking for first-order logic is fixed-parameter tractable on further interesting classes of graphs such as graphs of bounded degree [17], planar graphs [10], and more generally locally tree-decomposable classes of structures [10]. Theorem 1(2) does *not* imply lower bounds for the parameter dependence here. The reason for that is a peculiar detail in the encoding of words by relational structures. The standard encoding includes the linear order of the letters in a word as an explicit relation of the structure. If we omit the order and just include a successor relation, Theorem 1(1) still holds, because the order is definable in monadic second-order logic. However, the order is not definable in first-order logic, and Theorem 1(2) does not extend to words without order. Indeed, we give a model-checking algorithm for first-order logic on words without order, and more generally on structures of degree 2, with a running time $2^{2^{O(k)}} \cdot n$, that is, with a doubly exponential parameter dependence. We also give a model-checking algorithm for first-order logic on structures of bounded degree $d \geq 3$ with a triply exponential parameter dependence. We match these upper bounds by corresponding lower bounds:

Theorem 2. *Assume that* W$[1] \neq$ FPT.

(1) There is no model-checking algorithm for first-order logic on the class of words without order whose running time is bounded by

$$2^{2^{o(k)}} \cdot p(n),$$

for any polynomial p.

(2) There is no model-checking algorithm for first-order logic on the class of binary trees whose running time is bounded by

$$2^{2^{2^{o(k)}}} \cdot p(n),$$

for any polynomial p.

Again, k denotes the size of the input sentence and n the size of the input structure.

Part (2) of this theorem also implies a triply exponential lower bound for the parameter dependence of first-order model-checking on planar graphs and graphs of bounded tree-width.

2. Preliminaries

A *vocabulary* is a finite set of relation, function, and constant symbols. Each relation and function symbol has an *arity*. τ always denotes a vocabulary. A *structure* $\mathcal{A}$ of vocabulary τ, or τ-structure, consists of a set A called the universe, and an interpretation $T^{\mathcal{A}}$ of each symbol $T \in \tau$: Relation symbols and function symbols are interpreted by relations and functions on A of the appropriate arity, and constant symbols are interpreted by elements of A. *All structures considered in this paper are assumed to have a finite universe*. The *reduct* of a τ-structure $\mathcal{A}$ to a vocabulary $\tau' \subseteq \tau$ is the τ'-structure with the same universe as $\mathcal{A}$ and the same interpretation of all symbols in τ'. An expansion of a structure $\mathcal{A}$ is a structure $\mathcal{A}'$ such that $\mathcal{A}$ is a reduct of $\mathcal{A}'$. In particular, if $\mathcal{A}$ is a structure and $a \in A$, then by $(\mathcal{A}, a)$ we denote the expansion of $\mathcal{A}$ by the constant a. We write $\mathcal{A} \cong \mathcal{B}$ to denote that structures $\mathcal{A}$ and $\mathcal{B}$ are isomorphic.

Let Σ be a finite alphabet. We let $\tau(\Sigma)$ be the vocabulary consisting of a binary relation symbol $\leq$, a unary function symbol S, two constant symbols 'min' and 'max', and a unary relation symbol P_s for every $s \in \Sigma$. A *word structure* over Σ is a $\tau(\Sigma)$-structure $\mathcal{W}$ with the following properties:

- $\leq^{\mathcal{W}}$ is a linear order of W, $\min^{\mathcal{W}}$ and $\max^{\mathcal{W}}$ are the minimum and maximum element of $\leq^{\mathcal{W}}$, and $S^{\mathcal{W}}$ is the successor function associated with $\leq^{\mathcal{W}}$, where we let $S^{\mathcal{W}}(\max^{\mathcal{W}}) = \max^{\mathcal{W}}$.
- For every $a \in W$ there exists precisely one $s \in \Sigma$ such that $a \in P_s^{\mathcal{W}}$.

We refer to elements $a \in W$ as the *positions* in the word (structure) and, for every position $a \in W$, to the unique s such that $a \in P_s^{\mathcal{W}}$ as the *letter at* a.

It is obvious how to associate a word from the class Σ^* of all words over Σ with every word structure over Σ and, conversely, how to associate an up to isomorphism unique word structure with every word in Σ^*. We identify words with the corresponding word structures and write $\mathcal{W} \in \Sigma^*$ to refer both to the word and the structure.

The class of all words (or word structures) over some alphabet is denoted by $\mathbb{W}$. The length of a word $\mathcal{W}$ is denoted by $|\mathcal{W}|$.

A *subword* of a word $\mathcal{W} = s_0 \dots s_{n-1} \in \Sigma^*$ is either the empty word or a word $s_i \dots s_j$ for some $i, j, 0 \leq i \leq j < n$. We write $\mathcal{V} \sqsubseteq \mathcal{W}$ to denote that $\mathcal{V}$ is a subword of $\mathcal{W}$.

We assume that the reader is familiar with propositional logic, first-order logic FO and monadic second-order logic MSO (see, for example, [8]). If θ is a formula of propositional logic and α is a truth-value assignment to the variables of θ, then we write $\alpha \models \theta$ to denote that α satisfies θ. Similarly, if $\varphi(x_1, \dots, x_k)$ is a first-order or monadic second-order formula with free variables $x_1, \dots, x_k$, $\mathcal{A}$ is a structure, and $a_1 \dots, a_k \in A$, then we write $\mathcal{A} \models \varphi(a_1, \dots, a_k)$ to denote that $\mathcal{A}$ satisfies φ if the variables $x_1, \dots, x_k$ are interpreted by $a_1, \dots, a_k$, respectively. A *sentence* is a formula without free variables. The *quantifier-rank* of a formula φ, that is, the maximum number of nested quantifiers in φ, is denoted by $\mathrm{qr}(\varphi)$.

The *model-checking problem* for a logic L on a class C of structures, denoted by $\mathrm{MC}(\mathrm{L}, \mathrm{C})$, is the following decision problem:

Input:	Structure $\mathcal{A} \in$ C, sentence $\varphi \in$ L.
Problem:	Decide if $\mathcal{A} \models \varphi$.

We fix a reasonable encoding of structures and formulas by words over $\{0,1\}$. We denote the length of the encoding of a structure $\mathcal{A}$ by $||\mathcal{A}||$ and the length of the encoding of a formula φ by $||\varphi||$. Note that $||\mathcal{A}||$ can be considerably larger than the cardinality $|A|$ of the universe of $\mathcal{A}$. When reasoning about model-checking problems, we usually use n to denote the size $||\mathcal{A}||$ of the input structure and k to denote the size $||\varphi||$ of the input sentence.

It is well-known that if we are interested in the complexity of first-order or monadic-second order model-checking on words, the alphabet is inessential. This can be phrased as follows:

Fact 3. *Let* $\mathrm{L} \in \{\mathrm{FO}, \mathrm{MSO}\}$. *Then there is a linear time algorithm that, given a sentence* $\varphi \in \mathrm{L}$ *and a word* $\mathcal{W} \in \mathbb{W}$*, computes a sentence* $\varphi' \in \mathrm{L}$ *of vocabulary* $\tau(\{0,1\})$ *and a word* $\mathcal{W}' \in \{0,1\}^*$ *such that* $||\varphi'|| \in O(||\varphi||)$, $||\mathcal{W}'|| \in O(||\mathcal{W}||)$, *and* $\big(\mathcal{W} \models \varphi \iff \mathcal{W}' \models \varphi'\big)$.

$\mathbb{N}$ denotes the set of natural numbers (including 0). For all $n, i \in \mathbb{N}$ we let $\mathrm{bit}(i,n)$ denote the ith bit in the binary representation of n. (Here we count the lowest priority bit as the 0th bit.) lg denotes the base-2 logarithm, and, for $i \in \mathbb{N}$, $\lg^{(i)}$ denotes the i-fold logarithm. More formally, $\lg^{(i)}$ is defined by $\lg^{(0)}(n) = n$ and $\lg^{(i+1)}(n) = \lg \lg^{(i)}(n)$.

We define the *tower function* $T : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ by $T(0,r) = r$ and $T(h+1,r) = 2^{T(h,r)}$ for all $h \in \mathbb{N}$, $r \in \mathbb{R}$. Thus $T(h,r)$ is a tower of 2s of height h with an r sitting on top. Observe that for all $n, h \in \mathbb{N}$ with $n \geq 1$ we have $T(h, \lg^{(h)} n) = n$.

3. Succinct encodings

We introduce a sequence of encodings μ_h, for $h \geq 1$, of natural numbers by words over certain finite alphabets. They are more and more "succinct" not in the sense that they are shorter and shorter, but in the sense that they can be decoded by shorter and shorter first-order formulas. Lemma 6 is the key result of this section.

For all $h \geq 1$ we let $\Sigma_h = \{0, 1, [_1,]_1, \ldots, [_h,]_h\}$. We define $L : \mathbb{N} \to \mathbb{N}$ by $L(0) = 0$, $L(1) = 1$, $L(n) = \lfloor \lg(n-1) \rfloor + 1$ for $n \geq 2$. Note that for $n \geq 1$, $L(n)$ is precisely the length of the binary representation of $n-1$.

We are now ready to define our encodings $\mu_h : \mathbb{N} \to \Sigma_h^*$, for $h \geq 1$. We let $\mu_1(0) = [_1]_1$ and for $n \geq 1$

$$\mu_1(n) = [_1 \, \mathrm{bit}(0, n-1) \, \mathrm{bit}(1, n-1) \, \ldots \, \mathrm{bit}(L(n)-1, n-1) \,]_1$$

for $n \geq 1$. For $h \geq 2$, we let $\mu_h(0) = [_h \,]_h$ and

$$\mu_h(n) = [_h \, \mu_{h-1}(0) \, \mathrm{bit}(0, n-1) \, \mu_{h-1}(1) \, \mathrm{bit}(1, n-1) \, \ldots \, \mu_{h-1}(L(n)-1) \, \mathrm{bit}(L(n)-1, n-1) \,]_h \, .$$

Lemma 4. $$|\mu_h(n)| \in O(h \cdot \lg^2 n).$$

Proof: We define functions $L_i : \mathbb{N} \to \mathbb{N}$ as follows: $L_1(n) = L(n)$ for all $n \in \mathbb{N}$ and $L_i(n) = L_{i-1}(L(n))$ for all $i, n \in \mathbb{N}$ with $i \geq 2$. Moreover, we define $P_i : \mathbb{N} \to \mathbb{N}$ for $i \geq 1$ by

$$P_i(n) = \prod_{j=1}^{i} L_j(n).$$

Observe that for all $i \geq 2$ and $n \geq 1$ we have $P_i(n) = L(n) \cdot P_{i-1}(L(n))$.

We first prove, by induction on $h \geq 1$, that for all $n \geq 1$,

$$|\mu_h(n)| \leq 4h \cdot P_h(n). \tag{1}$$

We have $\mu_1(n) = 2 + L(n) \leq 4L(n) = 4P_1(n)$, so (1) is true for $h = 1$. Let $h \geq 2$ an suppose that (1) holds for $h - 1$. Then

$$\begin{aligned}
|\mu_h(n)| &= 2 + L(n) + \sum_{i=0}^{L(n)-1} |\mu_{h-1}(i)| \\
&= 2 + L(n) + 2 + \sum_{i=1}^{L(n)-1} |\mu_{h-1}(i)| \\
&\leq 4 + L(n) + \sum_{i=1}^{L(n)-1} 4(h-1) \cdot P_{h-1}(i) \\
&\leq 4 + L(n) + 4(L(n)-1) \cdot (h-1) \cdot P_{h-1}(L(n)) \\
&\leq L(n) + (h-1) \cdot L(n) \cdot P_{h-1}(L(n)) \\
&\leq L(n) + (h-1) \cdot P_h(n) \\
&\leq 4h \cdot P_h(n).
\end{aligned}$$

This proves (1)

It remains to prove that $P_h(n) \in O(L(n)^2)$, independently of h. Since $L(L(n)) \in O(\lg \lg n)$ and $L(n) \in \Omega(\lg n)$, there is an n_0 such that for all $n \geq n_0$ we have

$$L(L(n))^2 \leq L(n).$$

Note that $P = \{P_h(m) \mid m < n_0, h \geq 1\}$ is a finite set and let $c = \max(P)$.

We prove that $P_h(n) \leq c \cdot L(n)^2$ by induction on $h \geq 1$. Since $P_1(n) = L(n)$, this statement is true for $h = 1$. For $h \geq 2$, we have $P_h(n) = L(n) \cdot P_{h-1}(L(n))$. If $L(n) < n_0$, we have $P_{h-1}(L(n)) \leq c$ and thus $P_h(n) \leq cL(n)$. If $L(n) \geq n_0$, we have $L(L(n))^2 \leq L(n)$. By induction hypothesis, $P_{h-1}(L(n)) \leq c \cdot L(L(n))^2$. Thus

$$P_h(n) = L(n) \cdot P_{h-1}(L(n)) \leq L(n) \cdot c \cdot L(L(n))^2 \leq c \cdot L(n)^2.$$

□

Lemma 5. *There is an algorithm that, given $h, n \in \mathbb{N}$, computes $\mu_h(n)$ in time $O(|\mu_h(n)|) = O(h \cdot \lg^2 n)$.*

Proof: The algorithm computes $\mu_h(n)$ in a straightforward recursive manner. We get the following recurrence for the running time $R(h, n)$:

$$R(h, n) \leq O(L(n)) + \sum_{i=0}^{L(n)} R(h-1, L(i)).$$

This recurrence is very similar to the one we obtained in the proof of Lemma 4 and can easily be solved using the same methods. □

Observe that for all $m \geq 1$ we have

$$2^m = \max\{n \in \mathbb{N} \mid L(n) \leq m\}.$$

Recall that $T(h, \ell)$ is a tower of 2s of height h with an ℓ on top. Thus, in particular, for all $h, \ell \geq 1$ we have

$$T(h, \ell) = \max\{n \in \mathbb{N} \mid L(n) \leq T(h-1, \ell)\}. \tag{2}$$

Lemma 6. *Let $h \geq 1, \ell \geq 0$ and let $\Sigma \supseteq \Sigma_h$. There is a first-order formula $\chi_{h,\ell}(x, y)$ of vocabulary $\tau(\Sigma_h)$ and size $O(h + \ell)$ such that for all words $\mathcal{W} \in \Sigma^*$, $a, b \in W$, and $m, n \in \{0, \ldots, T(h, \ell)\}$ the following holds:*

If a is the first position of a subword $\mathcal{U} \sqsubseteq \mathcal{W}$ with $\mathcal{U} \cong \mu_h(m)$ and b is the first position of a subword $\mathcal{V} \sqsubseteq \mathcal{W}$ with $\mathcal{V} \cong \mu_h(n)$, then

$$\mathcal{W} \models \chi_{h,\ell}(a,b) \iff m = n.$$

Furthermore, the formula $\chi_{h,\ell}$ can be computed from h and ℓ in time $O(h+\ell)$.

Proof: Let $h = 1$. Recall that the μ_1-encoding of an integer $p \geq 1$ is just the binary encoding of $p-1$ enclosed in $[_1$, $]_1$. Hence to say that x and y are μ_1-encodings of the same numbers, we have to say that for all pairs $x+i, y+i$ of corresponding positions between x resp. y and the next closing $]_1$, there are the same letters at $x+i$ and $y+i$. For numbers p in $\{0,\ldots,T(1,\ell)\}$, there are at most $L(p) \leq \ell$ positions to be investigated. To express this, we let

$$\begin{aligned}\chi_{1,\ell}(x,y) = \exists x_1 \ldots \exists x_\ell \exists y_1 \ldots \exists y_\ell \ \Big(& Sx = x_1 \wedge \textstyle\bigwedge_{i=1}^{\ell-1} \big((P_{]_1} x_i \wedge x_i = x_{i+1}) \vee (\neg P_{]_1} x_i \wedge Sx_i = x_{i+1})\big) \\ & \wedge Sy = y_1 \wedge \textstyle\bigwedge_{i=1}^{\ell-1} \big((P_{]_1} y_i \wedge y_i = y_{i+1}) \vee (\neg P_{]_1} y_i \wedge Sy_i = y_{i+1})\big) \\ & \wedge \textstyle\bigwedge_{i=1}^{\ell} \big((P_0 x_i \leftrightarrow P_0 y_i) \wedge (P_1 x_i \leftrightarrow P_1 y_i)\big)\Big).\end{aligned}$$

Now let $h \geq 2$ and suppose that we have already defined $\chi_{h-1,\ell}(x,y)$. It will be convenient to have the following auxiliary formulas available:

$$\begin{aligned}\chi^h_{\text{int}}(x,y) &= x < y \wedge \forall z\big((x < z \wedge z \leq y) \rightarrow \neg P_{]_h} z\big), \\ \chi^h_{\text{last}}(x,y) &= x < y \wedge P_{]_h} y \wedge \forall z\big((x < z \wedge z < y) \rightarrow \neg P_{]_h} z\big)\end{aligned}$$

Intuitively, $\chi^h_{\text{int}}(x,y)$ says that y is in the interior of the subword of the form $\mu_h(p)$ starting at x and $\chi^h_{\text{last}}(x,y)$ says that y is the last position of the subword of the form $\mu_h(p)$ starting at x — provided, such a subword indeed starts at x.

To say that the subwords starting at x and y are μ_h-encodings of the same numbers, we have to say that for all positions w between x and the next closing $]_h$ and all positions z between y and the next enclosing closing $]_h$, if w and z are first positions of subwords isomorphic to $\mu_{h-1}(q)$ for some $q \in \mathbb{N}$, then the positions following these two subwords are either both 1s or both 0s. For all subwords of $\mu_h(p)$ of the form $\mu_{h-1}(q)$ we have $q \in \{0,\ldots,L(p)\}$. In order to apply the formula $\chi_{h-1,\ell}$ to test equality of such subwords, we must have $q \leq L(p) \leq T(h-1,\ell)$. By (2), the last inequality holds for all $p \leq T(h,\ell)$. Thus for such p we can use the formula $\chi_{h-1,\ell}$ to test equality of subwords of $\mu_h(p)$ of the form $\mu_{h-1}(q)$. As a first approximation to our formula $\chi_{h,\ell}$, we let

$$\begin{aligned}\chi'_{h,\ell}(x,y) = \forall w\Big(\big(\chi^h_{\text{int}}(x,w) \wedge P_{]_{h-1}} w\big) \rightarrow \exists z\big(\chi^h_{\text{int}}(y,z) \wedge P_{]_{h-1}} z \wedge \chi_{h-1,\ell}(w,z)\big)\Big) \\ \wedge \forall z\Big(\big(\chi^h_{\text{int}}(y,z) \wedge P_{]_{h-1}} z\big) \rightarrow \exists w\big(\chi^h_{\text{int}}(x,w) \wedge P_{]_{h-1}} w \wedge \chi_{h-1,\ell}(w,z)\big)\Big) \\ \wedge \forall w \forall z\Big(\big(\chi^h_{\text{int}}(x,w) \wedge P_{]_{h-1}} w \wedge \chi^h_{\text{int}}(y,z) \wedge P_{]_{h-1}} z \wedge \chi_{h-1,\ell}(w,z)\big) \\ \rightarrow \exists w' \exists z' \big(\chi^{h-1}_{\text{last}}(w,w') \wedge \chi^{h-1}_{\text{last}}(z,z') \wedge (P_1 Sz' \leftrightarrow P_1 Sw')\big)\Big).\end{aligned}$$

The first line of this formula says that every subword of the form $\mu_{h-1}(q)$ in the subword of the form $\mu_h(p)$ starting at x also occurs in the subword of the form $\mu_h(p)$ starting at y. The second line says that every subword of the form $\mu_{h-1}(q)$ in the subword of the form $\mu_h(p)$ starting at y also occurs in the subword of the form $\mu_h(p)$ starting at x. The third and fourth line say that if w and z are the first positions of isomorphic subwords of the form $\mu_{h-1}(q)$, then they are either both followed by a '1' or both by a '0' (since the only two letters that can appear immediately after a subword $\mu_{h-1}(q)$ in a word $\mu_h(p)$ are '0' and '1').

This formula says what we want, but unfortunately it is too large to achieve the desired bounds. The problem is that there are three occurrences of the subformula $\chi_{h-1,\ell}(w,z)$. We we can easily overcome this problem. We let

$$\zeta(w,z) = \exists w' \exists z' \big(\chi^{h-1}_{\text{last}}(w,w') \wedge \chi^{h-1}_{\text{last}}(z,z') \wedge P_1 Sz' \leftrightarrow P_1 Sw'\big)$$

and

$$\begin{aligned}\chi_{h,\ell}(x,y) = \forall w \exists z \Big(& \big(\chi^h_{\text{int}}(x,w) \to \chi^h_{\text{int}}(y,z)\big) \\ & \wedge \big(\chi^h_{\text{int}}(y,w) \to \chi^h_{\text{int}}(x,z)\big) \\ & \wedge \big(P_{\lfloor_{h-1}} w \to P_{\lfloor_{h-1}} z\big) \\ & \wedge \big(\big((\chi^h_{\text{int}}(y,w) \vee \chi^h_{\text{int}}(x,w)) \wedge P_{\lfloor_{h-1}} w\big) \to \chi_{h-1,\ell}(w,z) \wedge \zeta(w,z)\big)\Big).\end{aligned}$$

It is not hard to see that $\chi_{h,\ell}(x,y)$ has the desired meaning.

Observing that $||\chi_{1,\ell}|| \in O(\ell)$ and that $||\chi_{h,\ell}|| = ||\chi_{h-1,\ell}|| + c$ for some constant c, we obtain the desired bound on the size of the formulas.

The fact that $\chi_{h,\ell}$ can be computed in linear time is immediate from the construction. □

4. Encodings of propositional formulas

In this section, we use our encodings μ_h of the natural numbers to define encodings of propositional formulas and assignments. We will also denote these encodings of formulas by μ_h.

A propositional formula is *t-normalised*, for a $t \geq 1$, if it is a conjunction of disjunctions of conjunctions ... of literals, with $(t-1)$ alternations between conjunctions and disjunctions. More formally, we let $\Gamma_0 = \Delta_0$ be the set of all literals. For $t \geq 1$, we let Γ_t be the set of all (finite) conjunctions of formulas in Δ_{t-1} and Δ_t the set of all (finite) disjunctions of formulas in Γ_{t-1}. Then a formula is *t-normalised* if it is in Γ_t. Note that, in particular, a formula is 2-normalised if, and only if, it is in conjunctive normal form. For $k \geq 1$, a formula is in *k-conjunctive normal form* if it is a conjunction of disjunctions, where each disjunction contains at most k-literals. We denote the class of all formulas in k-conjunctive normal form by kCNF.

We assume that propositional formulas only contain variables X_i, for $i \in \mathbb{N}$. For a set Θ of propositional formulas, we let $\Theta(n)$ denote the set of all formulas in Θ whose variables are among $X_0, \ldots, X_{n-1}$.

For all $h, t \geq 1$, we let $\Sigma_{h,t} = \Sigma_h \cup \Big\{+, -, \text{TRUE}, \text{FALSE}, \star, \langle, \rangle, \{_0, \}_0, \ldots, \{_t, \}_t\Big\}$, where Σ_h is taken from the previous section (cf. page 5).

We fix h and define the encoding $\mu_h : \Gamma_t \cup \Delta_t \to \Sigma^*_{h,t}$ by induction on t.

For $t = 0$ and a literal λ, we let

$$\mu_h(\lambda) = \begin{cases} \{_0\, \mu_h(i)\ +\ \}_0 & \text{if } \lambda = X_i \\ \{_0\, \mu_h(i)\ -\ \}_0 & \text{if } \lambda = \neg X_i \end{cases}$$

(for every $i \in \mathbb{N}$).

Assuming that we have already defined $\mu_h(\theta)$ for all $\theta \in \Gamma_{t-1} \cup \Delta_{t-1}$, let $\eta = \bigwedge_{i=0}^{m-1} \theta_i$, with $\theta_i \in \Delta_{t-1}$, or $\eta = \bigvee_{i=0}^{m-1} \theta_i$, with $\theta_i \in \Gamma_{t-1}$, be a formula in $\Gamma_t \cup \Delta_t$. We let

$$\mu_h(\eta) = \{_t\, \mu_h(\theta_0)\, \mu_h(\theta_1)\ \ldots\ \mu_h(\theta_{m-1})\, \}_t\,.$$

We also need to encode assignments. Let $A(n)$ denote the set of all assignments $\alpha : \{X_0, \ldots, X_{n-1}\} \to \{\text{TRUE}, \text{FALSE}\}$. We extend our encoding μ_h to assignments and pairs consisting of formulas and assignments. For an assignment $\alpha \in A(n)$, we let

$$\mu_h(\alpha) = \langle \mu_h(0)\, \alpha(X_0)\, \rangle\, \langle\, \mu_h(1)\, \alpha(X_1)\, \rangle\ \ldots\ \langle\, \mu_h(n-1)\, \alpha(X_{n-1})\, \rangle.$$

For a pair $(\theta, \alpha) \in (\Gamma_t(n) \cup \Delta_t(n)) \times A(n)$ we let $\mu_h(\theta, \alpha) = \mu_h(\theta)\, \mu_h(\alpha)$.

The following lemma is an immediate consequence of Lemma 4 and Lemma 5:

Lemma 7. *Let $h \in \mathbb{N}$ and $(\theta, \alpha) \in (\Gamma_t(n) \cup \Delta_t(n)) \times A(n)$. Then $|\mu_h(\theta, \alpha)| = O\big(h \cdot \lg^2 n \cdot (||\theta|| + n)\big)$ and there is an algorithm that computes $\mu_h(\theta, \alpha)$ in time $O\big(h \cdot \lg^2 n \cdot (||\theta|| + n)\big)$ (that is, linear in the size of the output).*

Lemma 8. *For all $h, \ell, t \in \mathbb{N}$ there is a first-order sentence $\varphi_{h,\ell,t}$ of size $O(h+\ell+t)$ such that for all $n \leq T(h,\ell)$ and $(\gamma, \alpha) \in \Gamma_t(n) \times A(n)$,*

$$\mu_h(\gamma, \alpha) \models \varphi_{h,\ell,t} \iff \alpha \models \gamma.$$

Furthermore, the formula $\varphi_{h,\ell,t}$ can be computed from h, ℓ, t in time $O(h+\ell+t)$.

Proof: Let $\chi_{h,\ell}(x,y)$ be the formula defined in Lemma 6. Recall that it says that the subwords of the form $\mu_h(m)$ and $\mu_h(n)$ starting at x, y, respectively, are identical — provided that such subwords start at x and y and that $n, m \leq T(h,\ell)$. Also recall the formula

$$\chi^h_{\text{last}}(x,y) = x < y \wedge P_{]_h} y \wedge \forall z\big((x < z \wedge z < y) \rightarrow \neg P_{]_h} z\big),$$

defined in the proof of Lemma 6, which says that y is the last position of the subword of the form $\mu_h(n)$ starting at x.

By induction on t, we define formulas $\psi^\Gamma_{h,\ell,t}(x)$ and $\psi^\Delta_{h,\ell,t}(x)$ such that for all $h, \ell, t, u, n \in \mathbb{N}$ with $n \leq T(h,\ell)$ and $u \geq t$ and for all formulas $\theta \in \Gamma_u(n) \cup \Delta_u(n)$, assignments $\alpha \in A(n)$, and positions a in the word $\mu_h(\theta, \alpha)$ we have:

(i) If the $\eta \in \Gamma_t \cup \Delta_t$ of the subword $\mu_h(\eta) \sqsubseteq \mu_h(\theta,\alpha)$ starting at a is in Γ_t, then $\mu_h(\theta,\alpha) \models \psi^\Gamma_{h,\ell,t}(a) \iff \alpha \models \eta$.

(ii) If the $\eta \in \Gamma_t \cup \Delta_t$ of the subword $\mu_h(\eta) \sqsubseteq \mu_h(\theta,\alpha)$ starting at a is in Δ_t, then $\mu_h(\theta,\alpha) \models \psi^\Delta_{h,\ell,t}(a) \iff \alpha \models \eta$.

For $t = 0$, we let

$$\psi^\Gamma_{h,\ell,0}(x) = \psi^\Delta_{h,\ell,0}(x) = \exists y \exists x' \exists y' \Big(P_{\langle} y \wedge \chi_{h,\ell}(Sx, Sy) \wedge \chi^h_{\text{last}}(Sx, x') \wedge \chi^h_{\text{last}}(Sy, y') \\ \wedge \big(P_+ Sx' \leftrightarrow P_{\text{TRUE}} Sy'\big)\Big)$$

Suppose that the encoding of the literal $(\neg)X_i$ starts at x. The formula $\psi^\Gamma_{h,\ell,0}(x)$ looks for a y such that the encoding of a pair $(j, \alpha(X_j))$ starts at y, then compares i and j, and if they are equal, checks that the symbol indicating the sign of the literal is '+' if, and only if, $\alpha(X_j) = \text{TRUE}$.

For $t \geq 1$, we let

$$\begin{aligned} \psi^\Gamma_{h,\ell,t}(x) &= \forall y\Big(\big(\forall z\big((x < z \wedge z \leq y) \rightarrow \neg P_{\}_t} z\big) \wedge P_{\{_{t-1}} y\big) \rightarrow \psi^\Delta_{h,\ell,t-1}(y)\Big), \\ \psi^\Delta_{h,\ell,t}(x) &= \exists y\Big(\forall z\big((x < z \wedge z \leq y) \rightarrow \neg P_{\}_t} z\big) \wedge P_{\{_{t-1}} y \wedge \psi^\Gamma_{h,\ell,t-1}(y)\Big). \end{aligned}$$

The subformula $\forall z\big((x < z \wedge z \leq y) \rightarrow \neg P_{\}_t} z\big)$ guarantees that y belongs to the same level-t subformula as x, and the subformula $P_{\{_{t-1}} y$ guarantees that a level-$(t-1)$ subformula starts at y.

Once we have defined $\psi^\Gamma_{h,\ell,t}$, we let $\varphi_{h,\ell,t} = \psi^\Gamma_{h,\ell,t}(\text{min})$; it follows from (i) and the definition of μ_h that this sentence says what it is supposed to say.

To see that $||\varphi_{h,\ell,t}|| \in O(h+\ell+t)$, observe that $\varphi_{h,\ell,0}$ has exactly one subformula of the form $\chi_{h,\ell}$ and that each level-t formula has precisely one level-$(t-1)$ subformula. Finally, $\varphi_{h,\ell,t}$ can be computed from h, ℓ, t in time $O(h+\ell+t)$ simply by following our inductive definition. □

The reader may have noted that we included the symbol '$\star$' in our alphabet, but have not used it so far. In the next section, we want to use the formula of Lemma 8 for satisfiability testing. Of course when doing this we will not be given an assignment in advance. However, it will be useful if we nevertheless provide the "infrastructure" for the assignment in our encoding. To do this, we simply replace all the TRUE and FALSE symbols by $\star$s: For every formula $\theta \in \Gamma_t(n) \cup \Delta_t(n)$, we let $\mu_h(\theta, \star) = \mu_h(\theta)\,\langle \mu_h(0)\star\rangle\,\langle \mu_h(1)\star\rangle\,\ldots\,\langle \mu_h(n-1)\star\rangle$.

Remark 9. Of course, the $\star$ we just introduced is completely redundant — we could as well use FALSE. Actually, our encodings have many other redundancies. In introducing them, we tried to make the encodings a bit more structured and readable.

5. Satisfiability testing through model-checking

In this section, we prove the two statements of Theorem 1.

Theorem 10. *Unless* PTIME $=$ NP*, there is no algorithm for* MC(MSO, $\mathbb{W}$) *whose running time is bounded by*

$$T(h,k)\cdot p(n),$$

for any $h \in \mathbb{N}$ *and polynomial* p*. Here* k *denotes the size of the input sentence and* n *the size of the input word.*

Proof: Suppose that there is an algorithm ***A*** for MC(MSO, $\mathbb{W}$) whose runtime is bounded by $T(h,k)\cdot p(n)$, for some $h \in \mathbb{N}$ and polynomial p.

We shall prove that the satisfiability problem for 2-normalised propositional formulas (that is, formulas in conjunctive normal form) is in polynomial time, which, by contradiction, proves the theorem. For all $\ell \in \mathbb{N}$, let

$$\widetilde{\varphi}_{h+1,\ell} = \exists X\Big(\forall x(Xx \to P_\star x) \wedge \varphi'_{h+1,\ell,2}\Big),$$

where $\varphi'_{h+1,\ell,2}$ is the formula obtained from the formula $\varphi_{h+1,\ell,2}$ of Lemma 8 by replacing the subformula $P_{\text{TRUE}}Sy'$, which is the only subformula that involves either P_{TRUE} or P_{FALSE}, by XSy'. Then for every $n' \le T(h+1,\ell)$ and $\gamma \in \Gamma_2(n')$,

$$\mu_{h+1}(\gamma,\star) \models \widetilde{\varphi}_{h+1,\ell} \iff \gamma \text{ is satisfiable.} \tag{3}$$

Consider the algorithm displayed in Figure 1, which decides if the input formula γ is satisfiable.

Input: $\gamma \in \Gamma_2(n')$

1. Compute $\mu_{h+1}(\gamma,\star)$
2. Compute $\ell = \lceil \lg^{(h+1)}(n') \rceil$.
3. Compute $\widetilde{\varphi}_{h+1,\ell}$
4. Check if $\mu_{h+1}(\gamma,\star) \models \widetilde{\varphi}_{h+1,\ell}$ using algorithm ***A***.

Figure 1.

The correctness of the algorithm follows from (3) and $n' = T(h+1, \lg^{(h+1)}(n')) \le T(h+1, \lceil \lg^{(h+1)}(n') \rceil)$. We claim that the runtime of the algorithm is bounded by $q(||\gamma||)$ for some polynomial q depending only on the fixed constant h.

Lines 1–3 of the algorithm can be implemented in time polynomial in $n', h, ||\gamma||$. By Lemma 7, using our assumption on the algorithm ***A***, Line 4 requires time

$$T(h, ||\widetilde{\varphi}_{h+1,\ell}||)\cdot p(|\mu_{h+1}(\gamma,\star)|) \le T(h, ||\widetilde{\varphi}_{h+1,\ell}||)\cdot p_1(n')\cdot p_2(||\gamma||),$$

for some polynomials p_1 and p_2. Observe that $||\widetilde{\varphi}_{h+1,\ell}|| \in O(h+\ell)$, that is, $||\widetilde{\varphi}_{h+1,\ell}|| \le c(h+\ell) \le c(h + \lg^{(h+1)}(n') + 1)$ for some constant c. Since

$$\lim_{m\to\infty} \frac{\lg\lg m}{\lg m} = 0,$$

there is an n_0 (depending on c, h) such that for all $n' \ge n_0$ we have

$$c(h + \lg^{(h+1)}(n') + 1) \le \lg^{(h)}(n').$$

Thus for $n' \geq n_0$ we have $T(h, ||\widetilde{\varphi}_{h+1,\ell}||) \leq T(h, \lg^{(h)}(n')) \leq n'$. This proves the polynomial time bound. □

To state and prove our second main result, we need a few preliminaries from parameterized complexity theory. A *parameterized problem* is a set $P \subseteq \Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. If $(x, k) \in \Sigma^* \times \mathbb{N}$ is an instance of a parameterized problem, we refer to x as the *input* and to k as the *parameter*. A parameterized problem $P \subseteq \Sigma^* \times \mathbb{N}$ is *fixed-parameter tractable* if there is a computable function $f : \mathbb{N} \to \mathbb{N}$, a polynomial p, and an algorithm that, given a pair $(x, k) \in \Sigma^* \times \mathbb{N}$, decides if $(x, k) \in P$ in time at most $f(k) \cdot p(|x|)$ steps. The class of all fixed-parameter tractable problems is denoted by FPT.

The *W-hierarchy* is a family of complexity classes $\mathrm{W}[t]$, for $t \geq 1$, where

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots .$$

It is conjectured that this hierarchy is strict, or at least, that $\mathrm{FPT} \neq \mathrm{W}[1]$. The classes of the W-hierarchy are defined via complete problems under suitable reductions. These complete problems are parameterizations of the satisfiability problem for propositional formulas. The *weight* of a truth value assignment for a set of propositional variables is the number of variables set to TRUE by this assignment. For a class Θ of propositional formulas, let the *weighted satisfiability problem* for Θ, denoted by $\mathrm{WSAT}(\Theta)$, be the following parameterized problem:

Input:	$\theta \in \Theta$.
Parameter:	$k' \in \mathbb{N}$.
Problem:	Decide if θ has a satisfying assignment of weight k'.

Downey and Fellows [5, 7] proved that for all $t \geq 2$ the problem $\mathrm{WSAT}(\Gamma_t)$ is complete for $\mathrm{W}[t]$ under parameterized many-one reductions. In [6], they proved that $\mathrm{WSAT}(\mathrm{2CNF})$ is complete for $\mathrm{W}[1]$. Without giving a definition of the reductions, we can phrase the most important consequences of these results as follows:

Theorem 11 (Downey and Fellows [5, 6, 7]). *(1) Let $t \geq 2$. Then if $\mathrm{WSAT}(\Gamma_t)$ is fixed-parameter tractable, $\mathrm{W}[t] = \mathrm{FPT}$.*

(2) If $\mathrm{WSAT}(\mathrm{2CNF})$ is fixed-parameter tractable, then $\mathrm{W}[1] = \mathrm{FPT}$.

We are now ready to prove our theorem:

Theorem 12. *Unless $\mathrm{W}[t] = \mathrm{FPT}$ for all $t \in \mathbb{N}$, there is no algorithm for $\mathrm{MC}(\mathrm{FO}, \mathbb{W})$ whose runtime is bounded by*

$$T(h, k) \cdot p(n),$$

for any $h \in \mathbb{N}$ and polynomial p. As usual, k denotes the size of the input sentence and n the size of the input word.

To prove this theorem, we will use the following alternative characterisation of fixed-parameter tractability. A parameterized problem $P \subseteq \Sigma \times \mathbb{N}$ is *eventually in polynomial time* if there is a computable function f and an algorithm, whose runtime is polynomial in $|x|$ that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ of P with $|x| \geq f(k)$ correctly decides if $(x, k) \in P$. (The behaviour of the algorithm on instances $(x, k) \in \Sigma^* \times \mathbb{N}$ with $|x| < f(k)$ is irrelevant.)

Lemma 13 (Flum and Grohe [9]). *A parameterized problem is fixed-parameter tractable if, and only if, it is computable and eventually in polynomial time.*

Proof of Theorem 12: Suppose that there is an algorithm $\boldsymbol{A}$ for $\mathrm{MC}(\mathrm{FO}, \mathbb{W})$ whose runtime is bounded by

$$T(h, k) \cdot p(n),$$

for some $h \in \mathbb{N}$ and polynomial p.

Let $t \geq 1$. We shall prove that $\mathrm{WSAT}(\Gamma_t)$ is in FPT. For all $h, \ell, k' \in \mathbb{N}$, let

$$\widetilde{\varphi}_{h+1,\ell,t,k'} = \exists x_1 \ldots \exists x_{k'} \Big(\bigwedge_{i=1}^{k'} P_\star x_i \wedge \bigwedge_{i=1}^{k'-1} x_i < x_{i+1} \wedge \varphi'_{h+1,\ell,t,k'} \Big),$$

where $\varphi'_{h+1,\ell,t,k'}$ is the formula obtained from $\varphi_{h+1,\ell,t}$ by replacing the subformula $P_{\text{TRUE}} Sy'$ by $\bigvee_{i=1}^{k'} Sy' = x_i$. Then for every $n' \leq T(h+1,\ell)$ and $\gamma \in \Gamma_t(n')$,

$$\mu_{h+1}(\gamma, \star) \models \widetilde{\varphi}_{h+1,\ell,t,k'} \iff \gamma \text{ has a satisfying assignment of weight } k'. \tag{4}$$

Consider the algorithm displayed in Figure 2.

Input: $\gamma \in \Gamma_t(n')$, parameter $k' \in \mathbb{N}$

1. Compute $\mu_{h+1}(\gamma, \star)$
2. Compute $\ell = \lceil \lg^{(h+1)}(n') \rceil$.
3. Compute $\widetilde{\varphi}_{h+1,\ell,t,k'}$
4. Check if $\mu_{h+1}(\gamma, \star) \models \widetilde{\varphi}_{h+1,\ell,t,k'}$ using algorithm $\boldsymbol{A}$.

Figure 2.

The correctness of the algorithm follows from (4) and $n' = T(h+1, \lg^{(h+1)}(n')) \leq T(h+1, \lceil \lg^{(h+1)}(n') \rceil)$. We claim that if n' is sufficiently large, then the runtime of the algorithm is bounded by $q(||\gamma||)$ for some polynomial q. More precisely, we claim that there is a polynomial q and an $n_0 \in \mathbb{N}$, which is computable from k', h, t, such that for $n' \geq n_0$ the runtime of the algorithm is bounded by $q(n')$. Since h and t are fixed and since $\mathrm{WSAT}(\Gamma_t)$ is computable, by Lemma 13 this implies that $\mathrm{WSAT}(\Gamma_t)$ is in FPT.

Lines 1–3 of the algorithm can be implemented in time polynomial in $n', h, t, ||\gamma||$. By our assumption on the algorithm $\boldsymbol{A}$, Line 4 requires time

$$T(h, ||\widetilde{\varphi}_{h+1,\ell,t,k'}||) \cdot p(n) = T(h, ||\widetilde{\varphi}_{h+1,\ell,t,k'}||) \cdot p_1(n') \cdot p_2(||\gamma||),$$

for polynomials p_1, p_2 since $n = |\mu_{h+1}(\gamma, \star)|$ is polynomial in n' and h. Observe that $||\widetilde{\varphi}_{h+1,\ell,t,k'}|| \in O(k' + h + t + \ell)$, that is, $||\widetilde{\varphi}_{h+1,\ell,t,k'}|| \leq c(k' + h + t + \ell) \leq c(k' + h + t + \lg^{(h+1)}(n') + 1$ for some constant c.

Using the same argument as in the proof of Theorem 10, we can now derive that there is a computable n_0 such that for all $n' \geq n_0$ we have

$$T(h, ||\widetilde{\varphi}_{h+1,\ell,t,k'}||) \leq T(h, \lg^{(h)}(n')) \leq n'.$$

This proves our claim that if n' is sufficiently large, then the runtime of the algorithm is bounded by $q(||\gamma||)$ for some polynomial q and thus the theorem. □

Remark 14. For readers familiar with least fixed-point logic, let us point out that with the same techniques it can be proved that there is no model-checking algorithm for *monadic least fixed-point* logic on words whose running time is bounded by $T(h,k) \cdot p(n)$, for any $h \in \mathbb{N}$ and polynomial p, *under the weaker assumption that* $\mathrm{W}[P] \neq \mathrm{FPT}$.

$\mathrm{W}[P]$ is a parameterized complexity class that contains $\mathrm{W}[t]$ for all $t \geq 1$. A complete problem for $\mathrm{W}[P]$ is the weighted satisfiability problem for arbitrary Boolean circuits.

6. Structures of bounded degree

In this section, we investigate the parameterized complexity of first-order model-checking over structures of bounded degree. Let $\mathcal{A}$ be a τ-structure for some vocabulary τ. We call two elements $a, b \in A$ *adjacent* if they are distinct and there is an $R \in \tau$, say, r-ary, and a tuple $a_1 \dots a_r \in R^{\mathcal{A}}$ such that $a, b \in \{a_1, \dots, a_r\}$. The *degree* of an element $a \in A$ in the structure $\mathcal{A}$ is the number of elements adjacent to a, and the degree of $\mathcal{A}$ is the maximum degree of its elements. For $d \geq 1$, we denote the class of all structures of degree at most d by $\mathbb{D}(d)$.

Theorem 15 (Seese [17]). *Let $d \geq 1$. Then there is a function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm solving* $\mathrm{MC}(\mathrm{FO}, \mathbb{D}(d))$ *in time $f(k) \cdot n$, where, as usual, k denotes the size of the input sentence and n the size of the input structure.*

It is quite easy to derive from Seese's proof a triply-exponential upper bound on f for a *non-uniform* version of this theorem, stating that for every fixed first-order sentence φ there is a triply exponential function f and an algorithm checking whether a given structure $\mathcal{A}$ of degree at most d satisfies φ. We shall prove a uniform version of this result, which has the additional benefit that our algorithm is quite simple. But the main result of this section is a doubly exponential lower bound.

6.1. Upper bounds. In this section we present a general algorithm for first-order model-checking, which, restricted to structures of bounded degree, will yield optimal upper time bounds.

The crucial idea, which has also been explored by Seese, is to use the locality of first-order logic. Without loss of generality we assume that vocabularies only contain relation and constant symbols. (Functions can easily be simulated by relations.) We need some additional notation. A *path* of length l is a sequence of vertices $a_0, \dots, a_l \in A$ such that $a_{i-1}, a_i, i = 1, \dots, l$ are adjacent in $\mathcal{A}$. The distance between two elements $a, b \in A$ of the universe is 0, if $a = b$ and r, if the shortest path between a and b has length r. Let $r \geq 1$ and $a \in A$. The *r-neighbourhood* of a in $\mathcal{A}$, denoted by $N_r^{\mathcal{A}}(a)$ is the set of $b \in A$ such that a, b have distance at most r. Let $\mathcal{N}_r^{\mathcal{A}}(a)$ denote the substructure induced by $\mathcal{A}$ on $N_r^{\mathcal{A}}(a)$. For elements a, b of a structure $\mathcal{A}$ we write $a \cong_r^{\mathcal{A}} b$ if there is an isomorphism from $\mathcal{N}_r^{\mathcal{A}}(a)$ to $\mathcal{N}_r^{\mathcal{A}}(b)$ that maps a to b.

Lemma 16 ([13, 15]). *For every first-order formula $\varphi(x)$ there is an $r \geq 1$ such that for every structure $\mathcal{A}$ and $a, b \in A$ we have $\big(a \cong_r^{\mathcal{A}} b \implies (\mathcal{A} \models \varphi(a) \iff \mathcal{A} \models \varphi(b))\big)$. Furthermore, r can be chosen to be $2^{\mathrm{qr}(\varphi)}$.*

Figure 3 displays a recursive model-checking algorithm for first-order sentences in prenex normal form that is based on Lemma 16. Since we can easily transform arbitrary first-order sentences into sentences in prenex normal form (algorithmically, this can be done in linear time), this also gives us an algorithm for arbitrary sentences.

Note that in the recursive calls `model-check`$(\psi(a),(\mathcal{A}, a))$ of the algorithm, we replace all occurrences of x in ψ by a new constant symbol which is interpreted by the element $a \in A$ and check if this new sentence holds in the expanded structure $(\mathcal{A}, a)$. The correctness of the algorithm follows from an easy induction on the structure of the input formula φ applying Lemma 16 in each step. Note that this algorithm works for arbitrary input structures $\mathcal{A}$.

Theorem 17. *The algorithm* `model-check` *(displayed in Figure 3) decides* $\mathrm{MC}(\mathrm{FO}, \mathbb{D}(2))$ *in time*

$$2^{2^{O(k)}} \cdot n,$$

and $\mathrm{MC}(\mathrm{FO}, \mathbb{D}(d))$ *for $d \geq 3$ in time*

$$2^{2^{\lg d \cdot 2^{O(k)}}} \cdot n,$$

where as usual k denotes the size of the input sentence and n the size of the input structure.

Proof: We denote the runtime of `model-check`$(\varphi,\mathcal{A})$ by $R(n, p, q)$, where $n = ||\mathcal{A}||$, $q = \mathrm{qr}(\varphi) \leq k$, and p is the size of the quantifier-free part of φ. Note that $p + q \leq k (= ||\varphi||)$. Let $r = r(q) = 2^q$,

$$s(q) := \max_{a \in A, \mathcal{A} \in \mathcal{C}} ||\mathcal{N}_r^{\mathcal{A}}(a)||,$$

`model-check`(φ,$\mathcal{A}$)

1. **if** φ is quantifier free **then**
2. **accept** if φ holds in $\mathcal{A}$ and **reject** otherwise.

In the following, assume that $\varphi(x) = Qx\ \psi(x)$ for some quantifier Q.

3. Compute $r := 2^{\mathrm{qr}(\varphi)}$
4. Compute a set $X \subseteq A$ of representatives of the equivalence classes of the relation $\cong_r^{\mathcal{A}}$.
5. Recursively call `model-check`($\psi(a)$,$(\mathcal{A}, a)$) for all $a \in X$.
6. **if** $\varphi = \exists x \psi(x)$ **then**
7. **accept** if at least one of the recursive calls accepts and **reject** otherwise.
8. **if** $\varphi = \forall x \psi(x)$ **then**
9. **accept** if all recursive calls accept and **reject** otherwise.

Figure 3.

the maximal size of a r-neighbourhood, and let $t(q)$ denote the number of equivalence classes of $\cong_r^{\mathcal{A}}$. Note that there exist upper bounds for $s(q)$ and $t(q)$ only depending on the degree of the input structure (and not on n or φ). Remember that the degree is constant for the classes under consideration.

Now consider the algorithm displayed in Figure 3. Line 1 only requires constant time. If Line 2 is executed, it requires time $O(p \cdot n)$, and the algorithm stops. Otherwise, it proceeds to Line 3, which can be executed in constant time. To execute Line 4, we maintain a list of pairs $(\mathcal{N}_r^{\mathcal{A}}(a), a)$ such that no induced substructure $(\mathcal{N}_r^{\mathcal{A}}(a), a)$ occurs twice. The size of this list never exceeds $t(q)$, hence for each a in turn, we simply compute the induced substructure, and look if it is already in the list. This requires time $O(n \cdot (f(s(q))t(q) + s(q)))$, if we denote the time to check isomorphism of structures of size m by $f(m)$. The loop in Lines 5–9 requires time

$$O(t(q)) + t(q) \cdot R(n, p, q-1).$$

Putting everything together, we obtain the following recurrence for R:

$$\begin{aligned} R(n,p,0) &\leq c_1 \cdot p \cdot n \\ R(n,p,q) &\leq c_2 \cdot n(f(s(q))t(q) + s(q)) + t(q)R(n,p,q-1) \qquad \text{(for } q \geq 1\text{)}, \end{aligned}$$

for suitable constants c_1, c_2. To solve this equation, we use the following simple lemma:

Lemma 18. *Let $F, g, h : \mathbb{N} \to \mathbb{N}$ such that*

$$\begin{aligned} F(0) &\leq g(0) \\ F(m) &\leq g(m) + h(m) \cdot F(m-1) \end{aligned}$$

for all $m \in \mathbb{N}$. Then

$$F(m) \leq \sum_{i=0}^{m} g(i) \cdot \prod_{j=i+1}^{m} h(j)$$

for all $m \in \mathbb{N}$.

The lemma can be proved by a straightforward induction on q.

Applied to our function R, the lemma yields

$$\begin{aligned} R(n,p,q) &\leq c_1 \cdot p \cdot n \cdot \prod_{j=1}^{q} t(j) + \sum_{i=1}^{q} c_2 \cdot n \cdot \big(f(s(i))t(i) + s(i)\big) \cdot \prod_{j=i+1}^{q} t(j) \\ &\leq \prod_{j=1}^{q} t(j) \Big(c_1 \cdot p \cdot n + \sum_{i=1}^{q} c_2 \cdot n \cdot \big(f(s(i))t(i) + s(i)\big) \Big) \end{aligned}$$

Degree 2: The size of an r-neighbourhood in a structure $\mathcal{A} \in \mathbb{D}(2)$ is at most $2r+1$. Thus

$$s(q) \leq 2^{O(q)} \leq 2^{O(k)}.$$

To give an upper bound on $t(q)$, we have to take into account the number u of symbols in the vocabulary. Since we only have to consider symbols that actually appear in φ, we can assume that $u \leq k$. Moreover, without loss of generality we can assume that the vocabulary only contains unary and binary relation symbols (because we are considering structures of degree 2).

Let us count the number of isomorphism types of an m-vertex structure $\mathcal{B}$ of degree 2 whose vocabulary contains u_1 unary relation symbols and u_2 binary relation symbols. The unary relations can take at most $2^{u_1 \cdot m}$ different values. There are at most m pairs of elements which can be connected by a binary relation, thus the binary relations can take at most $2^{u_2 \cdot m}$ different values. Thus the overall number of isomorphism types is bounded by $2^{(u_1+u_2)m}$.

Our r-neighbourhoods have size at most $2r+1$, so we obtain

$$t(q) \leq 2^{O(k \cdot r)} = 2^{O(k \cdot 2^q)}.$$

Thus

$$\prod_{j=1}^{q} t(j) \leq \prod_{j=1}^{q} 2^{O(k \cdot 2^j)} \leq 2^{O(k \sum_{j=1}^{q} 2^j)} \leq 2^{2^{O(k)}}.$$

Since isomorphism of structures of degree 2 can be decided in polynomial time, we obtain

$$\Big(c_1 \cdot p \cdot n + \sum_{i=1}^{q} c_2 \cdot n \cdot \big(f(s(i))t(i) + s(i)\big) \Big) \leq O(2^{2^{O(k)}} \cdot n)$$

and thus

$$R(n,p,q) \leq 2^{2^{O(k)}} \cdot n.$$

Degree at least 3: The computations are similar in this case, the only important difference being that an r-neighbourhood may be of size $\Theta(d^r)$ and thus doubly exponential in q, which yields a triply exponential bound for R. □

6.2. The lower bounds.

In this subsection we examine two classes of particularly simple structures of degree two and three, the class of *words without order* and the class of *trees with distinguished* 0- *and* 1*-successors*.

Formally, a word without order over an alphabet Σ is a reduct of a word over Σ to the vocabulary $\tau_S(\Sigma) = \tau(\Sigma) \setminus \{\leq\}$. We denote the class of all words without order by $\mathbb{S}$. *Since we will only consider words without order in the following, for simplicity we often just refer to them as words.*

In this section we will only work with the encoding μ_1 (recall the definition from Section 3), but we need a refined version of Lemma 6 for $h = 1$:

Lemma 19. *Let $\ell \geq 1$ and let $\Sigma \supseteq \Sigma_1$. There is a first-order formula $\chi_\ell(x,y)$ of vocabulary $\tau_S(\Sigma_1)$ and size $O(\ell)$ such that for all words without order $\mathcal{W} \in \Sigma^*$, $a, b \in W$, and $m, n \in \{0, \ldots, 2^{2^\ell}\}$ the following holds:*

If a is the first position of a subword $\mathcal{U} \sqsubseteq \mathcal{W}$ with $\mathcal{U} \cong \mu_1(m)$ and b is the first position of a subword $\mathcal{V} \sqsubseteq \mathcal{W}$ with $\mathcal{V} \cong \mu_1(n)$, then

$$\mathcal{W} \models \chi_\ell(a,b) \iff m = n.$$

Furthermore, the formula χ_ℓ can be computed from ℓ in time $O(\ell)$.

Note that Lemma 6 only provides a formula $\chi_{1,l}(x,y)$ that works for $m, n \leq 2^\ell$.

Before we prove the lemma, we define a few basic formulas and notations that we need in dealing with words without order. Let $\psi(x,y)$ be a formula. For a structure $\mathcal{A}$ and elements $a, b \in A$ we let

$$b -_\psi a = \begin{cases} 0 & \text{if } a = b \\ i & \text{if not } b -_\psi a < i \\ & \text{and there exists } a_0 = a, a_2, \ldots, a_i = b \text{ such that } \mathcal{A} \models \psi(a_i, a_{i+1}) \text{ for } 0 \leq i < i \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Lemma 20. *Let $\ell \geq 1$ and $\psi(x,y)$ a first-order formula.*

(1) There exists a first-order formula $\beta^\psi_\ell(x_1,x_2)$ of size $O(\ell)$ such that for every structure $\mathcal{A}$ and all $a_1, a_2 \in A$,

$$\mathcal{A} \models \beta^\psi_\ell(a_1,a_2) \iff a_2 -_\psi a_1 \leq 2^\ell.$$

(2) There exists a first-order formula $\delta^\psi_\ell(x_1,x_2,y_1,y_2)$ of size $O(\ell)$ such that for every structure $\mathcal{A}$ and all elements $a_1, a_2, b_1, b_2 \in A$,

$$\mathcal{A} \models \delta^\psi_\ell(a_1,a_2,b_1,b_2) \iff a_2 -_\psi a_1 \leq 2^\ell \wedge a_2 -_\psi a_1 = b_2 -_\psi b_1.$$

Proof: We only prove (2); the proof of (1) is similar, but simpler. We let

$$\delta^\psi_0(x_1,x_2,y_1,y_2) = \big(x_1 = x_2 \wedge y_1 = y_2\big) \vee \big(\neg x_1 = x_2 \wedge \neg y_1 = y_2 \wedge \psi(x_1,x_2) \wedge \psi(y_1,y_2)\big),$$

and for $\ell \geq 1$

$$\begin{aligned} \delta^\psi_\ell(x_1,x_2,y_1,y_2) = {} & \delta^\psi_0(x_1,x_2,y_1,y_2) \\ & \vee \exists x_3 \exists y_3 \forall x \forall x' \forall y \forall y' \Big(\\ & \qquad \big((x = x_1 \wedge x' = x_3 \wedge y = y_1 \wedge y' = y_3) \\ & \qquad \vee (x = x_3 \wedge x' = x_2 \wedge y = y_3 \wedge y' = y_2)\big) \to \delta^\psi_{\ell-1}(x,x',y,y')\Big). \end{aligned}$$

□

Proof of Lemma 19: We let $\psi(x,y) = (\neg P_{]_1} x \wedge Sx = y) \vee (P_{]_1} x \wedge x = y)$ and

$$\chi_\ell(x,y) = \forall x' \forall y' \Big(\delta^\psi_\ell(x,x',y,y') \to \big((P_0 x' \leftrightarrow P_0 y') \wedge (P_1 x' \leftrightarrow P_1 y')\big)\Big),$$

where δ^ψ_ℓ is taken from Lemma 20(2). □

Recall that 2CNF(n) denotes the set of all formulas in 2-conjunctive normal form whose variables are among $X_0, \ldots, X_{n-1}$ and that $A(n)$ denotes the set of all truth-value assignments to these variables. Recall further the encodings of propositional formulas introduced in Section 4.

Lemma 21. *For all $l \in \mathbb{N}$ there is a first-order sentence φ_l of size $O(l)$ such that for all $n \leq 2^{2^l}$ and $(\gamma, \alpha) \in \text{2CNF}(n) \times A(n)$ we have $\mu_1(\gamma, \alpha) \models \varphi_l \iff \alpha \models \gamma$. Furthermore, φ_l can be computed in time $O(l)$.*

We omit the proof, which is very similar to the proof of Lemma 8. The proof would not work for arbitrary formulas in conjunctive normal form, it is crucial that the disjunctions have bounded length.

We are now ready to prove the main result of this section (which is Theorem 2):

Theorem 22. *Unless* $\text{W}[1] = \text{FPT}$, *there is no algorithm for* $\text{MC}(\text{FO}, \mathbb{S})$ *whose running time is bounded by*

$$2^{2^{o(k)}} \cdot p(n),$$

for any polynomial p, where k denotes the size of the input sentence and n the size of the input word.

Proof: Essentially, we proceed as for word structures. Suppose that there is an algorithm $\boldsymbol{A}$ running in time $2^{2^{f(k)}} \cdot p(n)$, for some polynomial p and a function $f(k) \in o(k)$. We shall prove that $\text{WSAT}[\text{2CNF}]$ is in FPT. For $l, k' \in \mathbb{N}$ let

$$\widetilde{\varphi}_{l,k'} := \exists x_1 \dots x_{k'} \big(\bigwedge_{i=1}^{k'} P_* x_i \bigwedge_{1 \leq i < j \leq k'} x_i \neq x_j \wedge \varphi'_{l,k'} \big),$$

where $\varphi'_{l,k'}$ is obtained from φ_l by replacing each atom of the form $P_{\text{TRUE}} t$ by $\bigvee_{i=1}^{k'} t = x_i$ and $P_{\text{FALSE}} t$ by $\bigwedge_{i=1}^{k'} t \neq x_i$, where t denotes an arbitrary term. Since the number of atoms we have to replace is independent of k' and ℓ, we have $||\widetilde{\varphi}_{l,k'}|| = O(l + {k'}^2)$. Observe that, using the previous lemma, for every $n' \leq 2^{2^l}$ and $\gamma \in \text{2CNF}(n')$ we have

$$\mu_1(\gamma, *) \models \widetilde{\varphi}_{l,k'} \iff \gamma \text{ has a satisfying assignment of size } k'.$$

The algorithm deciding k'-satisfiability of 2CNF is displayed as Figure 4.

Input: $\gamma \in \text{2CNF}(n')$, parameter $k' \in \mathbb{N}$

1. Compute $\mu_1(\gamma, \star)$
2. Compute $\ell = \lceil \lg \lg n' \rceil$.
3. Compute $\widetilde{\varphi}_{l,k'}$
4. Check if $\mu_1(\gamma, \star) \models \widetilde{\varphi}_{\ell,k'}$ using algorithm $\boldsymbol{A}$.

Figure 4.

The correctness of this algorithm is easy to see. We will show that there is a computable $n_0 \in \mathbb{N}$ and a polynomial q such that for all $n' \geq n_0$ the runtime is bounded by $q(n')$. By Lemma 13 and the fact that the problem is computable at all, this implies that $\text{WSAT}[\text{2CNF}]$ is in FPT.

Let $\gamma \in \text{2CNF}(n')$ be the input. Lines 1–3 can be done in time polynomial in n'. The crucial part is Line 4. By the assumption on algorithm $\boldsymbol{A}$ this line requires time

$$2^{2^{f(||\widetilde{\varphi}_{l,k'}||)}} \cdot p(n),$$

where $n = |\mu_1(\gamma, \star)| = O(\lg^2 n' \cdot (||\gamma|| + n'))$. By the previous remark, $||\widetilde{\varphi}_{l,k'}|| \leq c \cdot ({k'}^2 + l) \leq c \cdot ({k'}^2 + \lg \lg n' + 1)$ for some constant c. Hence for sufficiently large n' we have $||\widetilde{\varphi}_{l,k'}|| \leq c' \lg \lg n'$, say,

for $c' = 2c$. Since $f(k) \in o(k)$, there is an n_0 such that for all $n' \geq n_0$ we have $f(c' \lg \lg n') \leq \lg \lg n'$ and thus

$$2^{2^{f(||\varphi_{l,k}||)}} \leq 2^{2^{f(c' \lg \lg n')}} \leq 2^{2^{\lg \lg n'}} \leq n'.$$

This gives us the desired upper bound on the runtime of our algorithm. □

Our last theorem matches the upper bound for model-checking over structures of bounded degree greater than 2 by a corresponding lower bound.

We need some additional terminology. We view *ordered binary trees* as $\{S_0, S_1\}$-structures $\mathcal{T}$, with $S_0^{\mathcal{T}}$ and $S_1^{\mathcal{T}}$ being the left child and right child relations. We allow nodes to only have one child. For a finite alphabet Σ, we let $\tau_T(\Sigma) = \{S_0, S_1\} \cup \{P_s | s \in \Sigma\}$, where P_s, for $s \in \Sigma$, is a unary relation symbol. An ordered binary tree *over* Σ is a $\tau_T(\Sigma)$-structure whose τ-reduct is an ordered binary tree. We denote the class of all orderd binary trees over some finite alphabet by $\mathbb{T}$. For a node a of a tree $\mathcal{T} \in \mathbb{T}$ and $d \geq 1$, the *depth d subtree below a* is the subtree of $\mathcal{T}$ whose nodes are all descendants of a of distance at most d from a.

To proceed as in the word cases, we will encode natural numbers by trees and provide "short" formulas allowing to compare "large" encoded numbers. For $\ell \in \mathbb{N}$, let $\mathcal{T}_\ell$ be the ordered binary tree with vertex set $\{0, \ldots, \ell\}$ and root 0 in which the children of i are $2i + 1$ and $2i + 2$. Recall that $L(n)$ denotes the length of the binary encoding of $n \in \mathbb{N}$. We let $\mu(n)$ be the ordered binary tree over $\{0, 1\}$ whose underlying tree is $\mathcal{T}_{L(n)}$ and in which, for $i = 0, 1$,

$$P_i^{\mathcal{T}(n)} = \{j \leq L(n) \mid \text{bit}(j, n) = i\}.$$

Figure 5 gives an example.

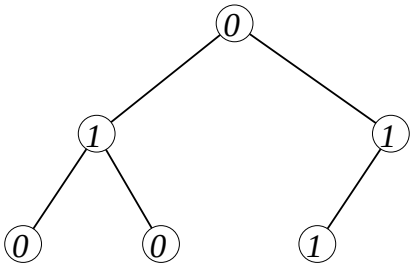

Figure 5. The tree $\mu(38)$

The next lemma is the key to the encoding and corresponds to the Lemmas 6 and 19.

Lemma 23. *Let $\ell \geq 1$. There is a formula $\chi_\ell(x, y)$ of vocabulary $\tau_T(\{0, 1\})$ of size $O(\ell)$ such that for all ordered binary trees $\mathcal{T} \in \mathbb{T}$, $a, b \in T$ and $m, n \in \{0, \ldots, 2^{2^{2^\ell}}\}$ the following holds:*

If the depth 2^ℓ subtree below a is isomorphic to $\mu(n)$ and the depth 2^ℓ subtree below b is isomorphic to $\mu(m)$ then

$$\mathcal{T} \models \chi_\ell(a, b) \iff m = n.$$

Furthermore $\chi_\ell(x, y)$ can be computed in time $O(\ell)$.

Proof: We construct a formula $\chi_\ell(x, y)$ characterising depth 2^ℓ subtrees up to isomorphism. This formula identifies binary encodings of length up to 2^{2^ℓ}, which proves the claim. We proceed as in the proof of

lemma 20. First, we say that to go from vertices x_1 to x_2 and from y_1 to y_2 we must follow the same sequence of S_0,S_1-successors. Let

$$\begin{aligned}\psi_0(x_1,x_2,y_1,y_2) := (&S_0x_1x_2 \wedge S_0y_1y_2)\\ \vee\, &(S_1x_1x_2 \wedge S_1y_1y_2)\\ \vee\, &(x_1 = x_2 \wedge y_1 = y_2).\end{aligned}$$

and for $l \geq 1$

$$\begin{aligned}\psi_l(x_1,x_2,y_1,y_2) := \exists x_3\exists y_3\forall x\forall x'\forall y\forall y'\big((&x_1 = x \wedge x_3 = x' \wedge y_1 = y \wedge y_3 = y')\\ \vee\, &(x_3 = x \wedge x_2 = x' \wedge y_3 = y \wedge y_2 = y') \rightarrow \psi_{l-1}(x,x',y,y')\big).\end{aligned}$$

Using this formula we let

$$\chi_l(x,y) := \forall x'\forall y'(\psi_l(x,x',y,y') \rightarrow ((P_1x' \leftrightarrow P_1y') \wedge (P_0x' \leftrightarrow P_0y')),$$

which is the sought formula. □

Now we proceed as before and encode formulas of 2CNF(n) for some n as an ordered binary tree over some alphabet Σ. For $\gamma \in$ 2CNF let $\mu(\alpha)$ be the binary tree $\mathcal{T}$ constructed as follows: let $\mathcal{W}$ be the word without order $\mu_1(\alpha)$, and consider $\mathcal{W}$ as a tree of S_1-successors without any S_0-successors. To get $\mathcal{T}$ we substitute each subword $\mathcal{U}$ of the form $\mu_1(m)$ by a single vertex v such that v's S_0-successor is the root of a copy of $\mu(m)$, while its S_1-successor is the first position after $\mathcal{U}$. v itself carries the new letter $\sharp$.

We extend the definition of μ to $(\gamma,\alpha) \in$ 2CNF $\times\, A(n)$ by applying the same substitution process to the part of the word corresponding to the assignment. Accordingly, we define $\mu(\gamma,\star)$. This encoding gives us the following lemma, whose proof is omitted since it resembles the proof of lemma 8 using the newly introduced encoding μ together with the decoding formulas $\chi_\ell(x,y)$. Observe that in these encodings we use the vocabulary $\Sigma := \{\sharp,0,1,*,+,-,\text{TRUE},\text{FALSE},\star,\langle,\rangle,\{_0,\}_0,\ldots,\{_2,\}_2\}$ ($[_0,]_0$ are no longer necessary).

Lemma 24. *For all $\ell \in \mathbb{N}$ there is a first-order sentence φ_ℓ of vocabulary $\tau_T(\Sigma)$ and size $O(l)$ such that for all $n \leq 2^{2^{2^\ell}}$ and $(\gamma,\alpha) \in$ 2CNF$(n) \times A(n)$ we have $\mu(\gamma,\alpha) \models \varphi_\ell \iff \alpha \models \gamma$. Furthermore, φ_ℓ can be computed in time $O(\ell)$.*

Now we are ready to state and prove the second lower bound of this section:

Theorem 25. *Unless* W$[1]$ = FPT, *there is no algorithm for* MC$(\text{FO},\mathbb{T})$ *whose running time is bounded by*

$$2^{2^{2^{o(k)}}} \cdot p(n),$$

for any polynomial p, where k denotes the size of the input sentence and n the size of the input tree.

Proof: Essentially, we proceed as for word structures. Suppose that there is an algorithm $\boldsymbol{A}$ running in time $2^{2^{2^{f(k)}}} \cdot p(n)$, for some polynomial p and a function $f(k) \in o(k)$. We shall prove that WSAT[2CNF] is in FPT. For $\ell,k' \in \mathbb{N}$ let

$$\widetilde{\varphi}_{\ell,k'} := \exists x_1 \ldots x_{k'}\big(\bigwedge_{i=1}^{k'} P_\star x_i \bigwedge_{1\leq i<j\leq k'} x_i \neq x_j \wedge \varphi'_{\ell,k'}\big),$$

where $\varphi'_{\ell,k'}$ is obtained from φ_ℓ from Lemma 24 by replacing each atom of the form $P_{\text{TRUE}}t$ by $\bigvee_{i=1}^{k'} t = x_i$ and $P_{\text{FALSE}}t$ by $\bigwedge_{i=1}^{k'} t \neq x_i$, where t denotes an arbitrary term. Since the number of atoms we have to

replace is independent of k' and ℓ, we have $||\widetilde{\varphi}_{\ell,k'}|| = O(\ell + {k'}^2)$. Observe that, using the previous lemma, for every $n' \leq 2^{2^{2^\ell}}$ and $\gamma \in \text{2CNF}(n')$ we have

$$\mu(\gamma, \star) \models \widetilde{\varphi}_{\ell,k'} \iff \gamma \text{ has a satisfying assignment of size } k'.$$

The algorithm deciding k'-satisfiability of 2CNF is displayed as Figure 6.

Input: $\gamma \in \text{2CNF}(n')$, parameter $k' \in \mathbb{N}$

1. Compute $\mu(\gamma, \star)$
2. Compute $\ell = \lceil \lg \lg \lg n' \rceil$.
3. Compute $\widetilde{\varphi}_{\ell,k'}$
4. Check if $\mu(\gamma, \star) \models \widetilde{\varphi}_{\ell,k'}$ using algorithm $\boldsymbol{A}$.

Figure 6.

The correctness of this algorithm is easy to see. We will show that there is a computable $n_0 \in \mathbb{N}$ (depending on k') and a polynomial q such that for all $n' \geq n_0$ the runtime is bounded by $q(n')$. By Lemma 13 and the fact that the problem is computable at all, this implies that WSAT[2CNF] is in FPT.

Let $\gamma \in \text{2CNF}(n')$ be the input. Lines 1–3 can be done in time polynomial in n'. The crucial part is Line 4. By the assumption on algorithm $\boldsymbol{A}$ this line requires time

$$2^{2^{2^{f(||\widetilde{\varphi}_{\ell,k'}||)}}} \cdot p(n),$$

where $n = ||\mu(\gamma, \star)|| = O((||\gamma|| + n')L(n'))$. By the previous remark, $||\widetilde{\varphi}_{\ell,k'}|| \leq c \cdot ({k'}^2 + \ell) \leq c \cdot ({k'}^2 + \lg \lg \lg n' + 1)$ for some constant c. Hence for sufficiently large n' we have $||\widetilde{\varphi}_{\ell,k'}|| \leq c' \lg \lg \lg n'$, say, for $c' = 2c$. Since $f(k) \in o(k)$, there is an n_0 such that for all $n' \geq n_0$ we have $f(c' \lg \lg \lg n') \leq \lg \lg \lg n'$ and thus

$$2^{2^{2^{f(||\widetilde{\varphi}_{\ell,k'}||)}}} \leq 2^{2^{2^{f(c' \lg \lg \lg n')}}} \leq 2^{2^{2^{\lg \lg \lg n'}}} \leq n'.$$

This gives us the desired upper bound on the runtime of our algorithm. □

7. Further Research

There is a significant gap between the lower bounds for model-checking on words provided by Theorem 1 and the upper bound $T(O(k), 1) \cdot n$ (a tower of 2s of height $O(k)$). It would be interesting to narrow this gap, maybe by proving that there is no $T(o(k), 1) \cdot p(n)$ algorithm for first-order or monadic second-order model-checking on words. Theorem 25 provides a triply exponential lower bound for first-order model-checking on planar graphs (because trees are planar). The best known upper bound is $T(O(k), 1) \cdot n$. Again it would be interesting to narrow the gap.

References

[1] K.A. Abrahamson, R.G. Downey, and M.R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogs. *Annals of pure and applied logic*, 73:235–276, 1995.

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

[3] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194–242. Elsevier Science Publishers, 1990.

[4] N.J. Cutland. *Computability*. Cambridge University Press, 1980.

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

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

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

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

[9] J. Flum and M. Grohe. Describing parameterized complexity classes. In H. Alt and A. Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *Lecture Notes in Computer Science*, pages 359–371. Springer-Verlag, 2002.

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

[11] M. Grohe. Descriptive and parameterized complexity. In J. Flum and M. Rodriguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, pages 14–31. Springer-Verlag, 1999.

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

[13] W. Hanf. Model-theoretic methods in the study of elementary logic. In J. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*, pages 132–145. North Holland, 1965.

[14] H. Kamp. *Tense Logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

[15] L. Libkin. Logics with counting and local properties. *ACM Transactions on Computational Logic*, 1:33–59, 2000.

[16] O. Lichtenstein and A. Pnueli. Finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth ACM Symposium on the Principles of Programming Languages*, pages 97–107, 1985.

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

[18] L.J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory*. PhD thesis, Department of Electrical Engineering, MIT, 1974.

[19] L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th ACM Symposium on Theory of Computing*, pages 1–9, 1973.

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