Noname manuscript No.
(will be inserted by the editor)

Parameterized Modal Satisfiability

Antonis Achilleos · Michael Lampis · Valia Mitsou

the date of receipt and acceptance should be inserted later

Abstract We investigate the parameterized computational complexity of the satisfiability problem for modal logic and attempt to pinpoint relevant structural parameters which cause the problem's combinatorial explosion, beyond the number of propositional variables v. To this end we study the modality depth, a natural measure which has appeared in the literature, and show that, even though modal satisfiability parameterized by v and the modality depth is FPT, the running time's dependence on the parameters is a tower of exponentials (unless P=NP). To overcome this limitation we propose possible alternative parameters, namely diamond dimension and modal width. We show fixed-parameter tractability results using these measures where the exponential dependence on the parameters is much milder (doubly and singly exponential respectively) than in the case of modality depth thus leading to FPT algorithms for modal satisfiability with much more reasonable running times. We also give lower bound arguments which prove that our algorithms cannot be improved significantly unless the Exponential Time Hypothesis fails.

1 Introduction

In this paper we consider the computational complexity of deciding formula satisfiability, for modal logics, focusing on the standard modal logic K. We attempt to present a new point of view on this important topic by making use of the parameterized complexity framework, which was pioneered by Downey and Fellows. Although the complexity of satisfiability for modal logic has been studied extensively in the past, to the best of our knowledge this is the first

A preliminary version of this paper has appeared in ICALP 2010.

Computer Science Department,
Graduate Center, City University of New York,
365 5th Ave New York, NY 10016 USA E-mail: antach@corelab.ntua.gr,E-mail: mlampis@gc.cuny.edu, E-mail: vmitsou@cs.gc.cuny.edu

time this has been done from an explicitly parameterized perspective. Moreover, the parameterized complexity of logic problems has been a fruitful field of research and we hope to extend this success to modal logic (some examples are the celebrated theorem of Courcelle [3] or the results of [8]; for an excellent survey on the interplay between logic, graph problems and parameterized complexity see [9]).

Modal logic is a family of systems of formal logic where the truth value of a sentence ϕ can be qualified by modality operators, usually denoted by $\Box$ and $\Diamond$. Depending on the specific modal logic and the application one considers, $\Box\phi$ and $\Diamond\phi$ can be informally read to mean, for example, "it is necessary that ϕ", or "it is known that ϕ" for $\Box$ and "it is possible that ϕ" for $\Diamond$. The fundamental normal modal logic system is known as K, while other common variations of this logic system include T, D, $\mathsf{S4}$, $\mathsf{S5}$. Modal logic systems provide a diverse universe of logics able to fit many modern applications in computer science (for example in AI or in game theory), making modal logic a widespread topic of research. The interested reader in the recent state of modal logic and its applications is directed to [1].

As in propositional logic, the satisfiability problem for modal logic is one of the most important and fundamental problems considered and many results are known about its (traditional) computational complexity. Ladner in [13] showed that satisfiability for K, T and $\mathsf{S4}$ is PSPACE-complete, while for $\mathsf{S5}$ the problem is NP-complete. Furthermore, in [10], Halpern shows that the problem remains PSPACE-complete when the formulae have at most one variable and in [2] it is shown that satisfiability for K and $\mathsf{K4}$ is PSPACE-complete even for formulae without any variables. In [12], Halpern and Rêgo showed that the negative introspection axiom is in an essential way what makes the difference between normal modal logics whose satisfiability problem is in NP and those for which it is PSPACE-complete. It should be noted that the satisfiability of propositional logic is a subcase of satisfiability for any normal modal logic, thus for any normal modal logic the problem is NP-hard. In this paper we will focus on the standard modal logic K. For an introduction to modal logic and its complexity see [11,5].

Traditional computational complexity theory attempts to characterize the complexity of a problem as a function of the input size n. The notion of parameterized complexity introduces to every hard problem a structural parameter k, which attempts to capture the aspect of the problem which causes its intractability. The central notion of tractability in this theory is called fixed-parameter tractability (FPT): an algorithm is called FPT if it runs in time $O(f(k) \cdot n^c)$, where f is any recursive function and c a constant. For an introduction to the vast area of parameterized complexity see [4,7].

Because the definition of FPT allows for any recursive function $f(k)$, fixed-parameter tractable problems can have complexities which depend on k in very different ways, ranging from sub-exponential to non-elementary. Thus, it is one of the main goals of parameterized complexity research to find the best possible $f(k)$ for every problem and this will be one of the main concerns of our work.

Our contribution

In this paper we study the complexity of modal satisfiability from a parameterized, or multi-variate, point of view. Just as parameterized complexity attempts to refine traditional complexity theory by more specifically identifying the aspects of an intractable problem which cause the problem's unavoidable combinatorial explosion, we attempt to identify some structural aspects of modal formulae which can have an impact on the solvability of satisfiability.

One natural parameter for the satisfiability problem (in any logic) is the number of propositional variables in the formula, which we denote by v. In propositional logic, when v is taken as a parameter, the propositional satisfiability problem trivially becomes fixed-parameter tractable. As was already mentioned, this does not generally hold in the case of satisfiability for modal logics where the problem is hard even for constant number of variables.

On the other hand since the satisfiability problem for modal logics is a generalization of the same problem for propositional logics, considering the modal satisfiability problem without bounding the number of variables or imposing some other propositional restriction on the formulae will result in an intractable problem. It would certainly be interesting to investigate modal satisfiability when certain structural propositional restrictions are placed (for example, we could say we are interested in formulae such that removing all modality symbols leaves a 2-CNF or a Horn formula, which are tractable cases of propositional satisfiability) but this goes beyond the scope of this work[1]. In this paper we will focus on strictly modal structural formula restrictions and therefore we will assume that the best way to make propositional satisfiability tractable is to restrict the number of variables. Informally, we could say that we are focusing on a case that is trivial for propositional logic, because we hope this will help us better understand how the addition of modalities affects the complexity of satisfiability. For our purposes the conclusion is that for modal satisfiability to become tractable, bounding v is necessary but not sufficient.

Motivated by the above we take the approach of a double parameterization: we investigate the complexity of satisfiability when v is considered a parameter and at the same time some other aspect contributing to the problem's complexity is identified and bounded.

We first study a natural notion of formula complexity called modality depth or modal depth. This complexity measure is already known in [10] (see also [15]) where in fact a fixed-parameter tractability result is shown when the problem is parameterized by the sum of v and the modality depth of the formula. However, since parameterized complexity was not well-known at the time, in [10] it is only pointed out that the problem is solvable in linear time for fixed values of the parameters, without mentioning how different values of v and the depth affect the running time. We address this by upper bounding the running time by an exponential tower of height equal to the modality depth of the formula. More importantly, we show a lower bound argument

[1] However, see [14] for related (non-parameterized) complexity results

which proves that even though the problem is FPT, this exponential tower in the running time cannot be avoided unless P=NP (Theorem 2). Our hardness proof follows an approach of encoding a propositional formula into a modal formula with very small modality depth. This draws a nice connection with previously known lower bound results of this form which also use a similar idea to prove the hardness of some model checking problems for first and second-order logic ([8] and the relevant chapter in [7]).

This result indicates that modal depth is unlikely to be a very useful parameter because even for formulae where the depth is very moderate the satisfiability problem is still very hard. This begs the natural question of whether there is a way to work around the lower bound of Theorem 2 by using another formula complexity measure in the place of modal depth. We show that this is indeed possible by introducing two alternative formula complexity notions.

Specifically, we define the notion of diamond dimension and show that satisfiability is FPT when parameterized by v and the diamond dimension and the dependence on the parameters is (only) doubly exponential. We then demonstrate a lower bound argument which proves that this dependence cannot be significantly improved unless the Exponential Time Hypothesis fails, that is, unless there exists an algorithm for n-variable 3-CNF-SAT running in time $2^{o(n)}$.

Then we define a measure called modal width and show that satisfiability is FPT when parameterized by v and the modal width and the dependence on the parameters is now just singly exponential.

Thus, our work shows that there exist many natural formula complexity parameters worth examining in the context of modal satisfiability and what's more that their complexity behavior can be vastly different and this could be an interesting field of study. Let us also note in passing that our results for modal width and depth directly apply also to satisfiability's dual problem, formula validity, since the validity of a formula can be solved by checking the satisfiability of its negation and every formula has the same width and depth as its negation. Our results for diamond dimension can also be extended for this problem by defining a dual "box dimension" measure, suitable for the validity problem.

2 Modal Logic

In this paper we study the language of modal logic. This language contains exactly the formulae that can be constructed using propositional variables, the standard propositional operators $\wedge, \vee, \neg$ (and the operators which can be defined using these, such as $\rightarrow$, $\leftrightarrow$) and the unary modality operators ($\Box$, $\Diamond$).

More specifically, the language of modal logic is defined recursively in the following way. We have a set of propositional variables, P. Any variable in P is a formula. Furthermore, if ϕ, ψ are formulae, then $(\phi \wedge \psi), (\phi \vee \psi), \neg\phi$ and $\Box\phi, \Diamond\phi$ are formulae of the language.

It is usually assumed that P is infinite, but for our purposes we do not need to assume anything about its cardinality. However, if $P = \emptyset$, we need to include $\bot$ (or $\top$) in our language in order to be able to form formulae. It is true that we do not need so many operators in our language and that there are many choices for more succinct list of initial symbols (ex. $\wedge$ and $\neg$; $\bot$ and $\rightarrow$, etc, with either $\Box$ or $\Diamond$), but for our purposes it is convenient to include all these.

In our language we do not include the constants $\bot$ and $\top$, for false and true, but we may use them to form formulas, as they can be considered shorthand for $x \wedge \neg x$ and $x \vee \neg x$ respectively, where $x \in P$.

Standard Kripke semantics are considered here: a Kripke frame is a pair (W, R) of a set of states W and an accessibility relation R between states. A Kripke frame together with a valuation V, which is a function that defines for each propositional variables the set of states where it is true, is called a Kripke structure.

In this paper we consider the system of modal logic usually denoted by K, where R is allowed to be an arbitrary relation between states. Other standard modal logics (e.g. T,D,S4) can be obtained by imposing various restrictions on R (e.g. if we only allow reflexive relations).

Given a Kripke structure $\mathcal{M} = (W, R, V)$, we define the relation $\models$ between states and formulae recursively on the structure of the formula:

$\mathcal{M}, s \models p$ if and only if $s \in V(p)$,
$\mathcal{M}, s \models \phi \wedge \psi$ if and only if $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$,
$\mathcal{M}, s \models \phi \vee \psi$ if and only if $\mathcal{M}, s \models \phi$ or $\mathcal{M}, s \models \psi$,
$\mathcal{M}, s \models \neg\phi$ if and only if $\mathcal{M}, s \not\models \phi$,
$\mathcal{M}, s \models \Box\phi$ if and only if for any $v \in W$, if sRs', then $\mathcal{M}, s' \models \phi$,
$\mathcal{M}, s \models \Diamond\phi$ if and only if there is some $s' \in W$, such that sRs' and $\mathcal{M}, s' \models \phi$,

where $p \in P$, $s \in W$ and ϕ, ψ are formulae.

When $\mathcal{M}, s \models \phi$, we say that ϕ is satisfied at s, or that ϕ is true at s and that ϕ is satisfied in $\mathcal{M}$, or that $\mathcal{M}$ is a model for ϕ. A formula ϕ is valid in a structure $\mathcal{M}$, if it is satisfied at all states s of the structure and we say that ϕ is valid if ϕ is valid in all structures.

The problem studied in this paper is *modal satisfiability* for K, that is, given a modal formula ϕ, does ϕ have model? The *modal validity* for K is the problem of determining whether a given modal formula is valid. Although we focus on satisfiability, the two problems are equivalent for modal logic, as any formula ϕ is satisfiable if and only if $\neg\phi$ is not valid.

In the following sections, three measures of formula complexity will be defined and we will study how they influence the difficulty of solving the modal satisfiability problem from a parameterized point of view.

3 Modal Depth

In this section we give the definition of modality depth. As we will see, a fixed-parameter tractability result can be obtained when satisfiability is parameterized by the number of propositional variables v and the modality depth of the input formula. This was first observed in [10], but in this section we more precisely bound the running time (in [10] it was simply noted that the running time is linear for constant depth and constant v with a hidden constant which "may be huge"). More importantly we show that the "huge constant" cannot be significantly improved by giving a hardness proof which shows that, if the running time of an algorithm for modal satisfiability is significantly less than an exponential tower of height equal to the modality depth, then P=NP.

Definition 1 The modality depth of a modal formula ϕ is defined inductively as follows:

- $\mathrm{md}(p) = 0$, if p is a propositional variable,
- $\mathrm{md}(\Diamond\phi) = \mathrm{md}(\Box\phi) = 1 + \mathrm{md}(\phi)$,
- $\mathrm{md}(\phi_1 \vee \phi_2) = \mathrm{md}(\phi_1 \wedge \phi_2) = \max\{\mathrm{md}(\phi_1), \mathrm{md}(\phi_2)\}$,
- $\mathrm{md}(\neg\phi) = \mathrm{md}(\phi)$

Note that, since for all ϕ we have $\mathrm{md}(\phi) = \mathrm{md}(\neg\phi)$ this implies that the results of this section, which we state in terms of the satisfiability problem, also apply to the validity prodblem, since deciding if some formula is valid is equivalent to deciding if its negation is satisfiable.

Theorem 1 ([10]) *Modal satisfiability for the logic* K *is FPT when parameterized by* v *and* $md(\phi)$*.*

Proof We define the d-type of a state s in a Kripke structure $\mathcal{M}$ to be the set $\{\phi \mid (\mathcal{M}, s) \models \phi \text{ and } \mathrm{md}(\phi) \leq d\}$. We will prove by induction on d that if we restrict ourselves to formulae with at most v variables then for any $d \geq 0$ there are at most $f_v(d)$ d-types, where f_v is the function recursively defined: $f_v(0) = 2^v$, $f_v(n+1) = 2^{f_v(n)+v}$.

For $d = 0$ If $\mathrm{md}(\phi) = 0$, then the formula is propositional, thus the 0-type of any state is directly defined by the set of propositional variables assigned true in the state. The number of all such possible sets of variables is $2^v = f_v(0)$.

For the case of $d+1$ The $(d+1)$-type of a state s depends on the assignment of the propositional variables in s and on the truth values of formulae of the forms $\Box\phi'$ and $\Diamond\phi'$, where $\mathrm{md}(\phi') \leq d$. Notice that these truth values depend only on the set of d-types of the accessible states from s. Thus the number of different $(d+1)$-types on a state s is $f_v(d+1) = 2^{f_v(d)+v}$.

Now, suppose that ϕ is a satisfiable formula of modality depth $d \geq 1$. We will show how to construct a Kripke structure of about $f_v(d-1)$ states to satisfy ϕ. To achieve this, for all $i \in \{0, 1, \ldots, d-1\}$ and for all i-types we will construct a state of that i-type, thus in total we will construct $\sum_{i=0}^{d-1} f_v(i) = O(f_v(d-1))$ states. To construct the $f_v(0) = 2^v$ states that give all the different 0-types we just construct 2^v states, each with a different valuation of the propositional variables. For the subsequent levels, to construct all the states for all the different $(i + 1)$-types we pick for each state a set of successor states out of the states that give us the different i-types and a valuation of the propositional variables. If ϕ is satisfiable, it must be satisfiable in this structure by adding a new state s, selecting a subset of the states that give us the different $(d - 1)$-types to be its successors and a valuation of the propositional variables in s. The number of combinations of all possible subsets of successors and all variable valuations is $f_v(d)$, so the problem is solvable in $O(f_v(d) \cdot f_v^2(d-1) \cdot |\phi|)$, because the structure has $O(f_v(d-1))$ states and thus size $O(f_v^2(d-1))$ and model checking can be performed in bilinear time (linear with respect to both $|\phi|$ and the size of the model).

□

Lower Bound

Let us now proceed to the main result of this section, which is that even though modal satisfiability is fixed-parameter tractable, the exponential tower in the running time cannot be avoided. Specifically, we will show that solving modal satisfiability parameterized by modality depth, even for constant v, requires a running time which is a tower of exponentials with height linear in the modality depth. We will prove this under the assumption that P≠NP, by reducing the problem of propositional satisfiability to our problem. Our proof follows ideas similar to those found in [8].

Suppose that we are given a propositional CNF formula ϕ_p with variables $x_1, \ldots, x_n$ and we need to check whether there exists a satisfying assignment for it. We will encode ϕ_p into a modal formula ϕ_m (the subscripts p and m stand for propositional and modal respectively) with small depth and a constant number of variables. In order to do so we inductively define a sequence of modal formulae.

– In order to encode the variables of ϕ_p we need some formulae to encode numbers (the indices of the variables). The modal formula v_i is defined inductively as follows[2]: $v_0 := \Box\bot$ and $v_n := \left(\bigwedge_{i:n_i=1} \Diamond v_i\right) \wedge \Box \left(\bigvee_{i:n_i=1} v_i\right)$ where by n_i we denote the i-th bit of n when n is written in binary and the least significant bit is numbered 0. So, for example $v_1 = \Diamond v_0 \wedge \Box v_0$, $v_2 = \Diamond v_1 \wedge \Box v_1$, $v_5 = \Diamond v_2 \wedge \Diamond v_0 \wedge \Box(v_2 \vee v_0)$ and so on. Observe that v_0 can only be true in a state with no successor states. Also, what is important is that these formulae allow us to encode very large numbers using only a very small modality depth and no variables (or just one variable if $\bot$ is considered short for $x \wedge \neg x$).

[2] We will use := to denote syntactic definitions of formulae and = to denote syntactic equality between formulae.

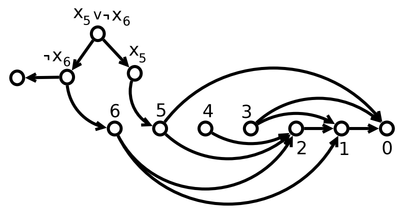

Fig. 1 A partial example, illustrating our construction for a specific clause. For the encoding of the clause $(x_5 \vee \neg x_6)$ we build the formula $\mathcal{C}(x_5 \vee \neg x_6)$ which holds in the state at the top of the depicted model.

- Next, we need to encode the literals of ϕ_p. The modal formula $\mathcal{L}(x_i)$ is defined as $\mathcal{L}(x_i) := \Diamond v_i \wedge \Box v_i$. The formula $\mathcal{L}(\neg x_i)$ is defined as $\mathcal{L}(\neg x_i) := \Diamond v_i \wedge \Diamond v_0 \wedge \Box (v_i \vee v_0)$.
- Now, to encode clauses we set $\mathcal{C}(l_1 \vee l_2 \vee \ldots \vee l_k) := \left(\bigwedge_{i=1}^{k} \Diamond \mathcal{L}(l_i)\right) \wedge \Box \left(\bigvee_{i=1}^{k} \mathcal{L}(l_i)\right)$.
- Finally, to encode the whole formula we use $\mathcal{F}(c_1 \wedge c_2 \wedge \ldots \wedge c_m) := \bigwedge_{i=1}^{m} \Diamond \mathcal{C}(c_i)$

So far we have described how to construct a modal formula $\mathcal{F}(\phi_p)$ from ϕ_p. $\mathcal{F}(\phi_p)$ encodes the structure of ϕ_p. Now we need to add two more ingredients: we must use a modal formula to describe that ϕ_p is satisfied by an assignment and that the assignment is consistent among clauses. We give two more formulae, $\mathcal{S}$ and $\mathcal{CA}(n)$, which play the previously described roles respectively:

- $\mathcal{S} := \Box\Diamond\left[((\Diamond v_0) \rightarrow (\Box\neg y)) \wedge ((\neg\Diamond v_0) \rightarrow (\Box y))\right]$, where we have introduced a single variable y.
- $\mathcal{CA}(n) := \bigwedge_{i=1}^{n} (\Diamond\Diamond\Diamond(y \wedge v_i) \leftrightarrow \neg\Diamond\Diamond\Diamond(\neg y \wedge v_i))$

Our full construction is, given a propositional CNF formula ϕ_p with n variables named $x_1, \ldots, x_n$, we create the modal formula $\phi_m := \mathcal{F}(\phi_p) \wedge \mathcal{S} \wedge \mathcal{CA}(n)$.

Lemma 1 *ϕ_p is satisfiable if and only if ϕ_m is satisfiable in* K.

Proof Suppose that ϕ_m is true at a state s of some Kripke structure. Then $\mathcal{CA}(n)$ is true at s therefore for each i we have either that $\Diamond\Diamond\Diamond(y \wedge v_i)$ is true at s or that $\Diamond\Diamond\Diamond(\neg y \wedge v_i)$ is true at s. From this we create a satisfying assignment: for those i for which the first holds we set $x_i = \top$ and for the rest $x_i = \bot$. We will show that this assignment satisfies ϕ_p.

Suppose that it does not satisfy ϕ_p, therefore there is some clause c_i which is not satisfied. However, since $\mathcal{F}(\phi_p)$ is true at s there exists a state p with sRp such that $\mathcal{C}(c_i)$ is true at p. In every successor state of p we have that $\mathcal{L}(l_j)$ is true for some literal l_j of c_i and there exists such a state for every literal of c_i. Also, in s we have that $\mathcal{S}$ is true, therefore in p we have $\Diamond\left[((\Diamond v_0) \rightarrow (\Box\neg y)) \wedge ((\neg\Diamond v_0) \rightarrow (\Box y))\right]$. Therefore, in some q such that pRq we have $((\Diamond v_0) \rightarrow (\Box\neg y)) \wedge ((\neg\Diamond v_0) \rightarrow (\Box y))$ and we also have that $\mathcal{L}(l_j)$ is true for some literal l_j of c_i. Suppose that l_j is a negated literal, that is $l_j = \neg x_k$. Then $\mathcal{L}(l_j) = \Diamond v_k \wedge \Diamond v_0 \wedge \Box(v_k \vee v_0)$. Therefore, since $\Diamond v_0$ is true at q this means that $\Box\neg y$ is true. Because $\Diamond v_k$ and $\Box\neg y$ are both true at q there exists an r such that qRr and $v_k \wedge \neg y$ is true at r. But then $\Diamond\Diamond\Diamond(v_k \wedge \neg y)$ is true at s which implies that our assignment gives the value *false* to x_k. Since c_i contains $\neg x_k$ it must be satisfied by our assignment, a contradiction. Similarly, if $l_j = x_k$ then $\mathcal{L}(l_j) = \Diamond v_k \wedge \Box v_k$. Clearly, v_0 and v_k cannot be true at the same state for $k > 0$ therefore in q we have $\neg\Diamond v_0$ which implies $\Box y$. Therefore in some r with qRr we have $y \wedge v_k$ which implies that our assignment sets x_k to *true* and since c_i has the literal x_k it must be satisfied.

The other direction is easier. We build a Kripke structure where for each v_i there exists a state such that v_i holds in that state. We start by introducing a state without successors, in which v_0 holds. Then, for each $i \in \{1, \ldots, n\}$ we add a state with appropriate transitions to states previously introduced so that v_i holds in that state (see Figure 1 for an example).

Note that each time we construct a new state and place the appropriate transitions so that v_i holds in that state, we know that no other v_j with $j \neq i$ can hold in that state. The reason is that, as follows from the definition of v_i, the formula $v_i \wedge v_j$ for $i \neq j$ is unsatisfiable. This, in turn, can be established by induction: first, $v_1 \wedge v_0$ is obviously unsatisfiable. Second, if for some $i > j$ we have $v_i \wedge v_j$ is true at some state of some model, then in some state accessible from it we will have $v_k \wedge v_l$, where k is the position of the most significant bit where i and j differ and $l \neq k$ is the position of some bit of j that is set to 1. Clearly, $k < i$ and $l < j$ so this contradicts the inductive hypothesis.

Now the completion of the Kripke structure so that ϕ_m is satisfied is straightforward. For every i with $1 \leq i \leq n$ we create two more states: the first has as its only successor the state where v_i is true. The other has two successors: the state where v_i is true and a state without successors (where v_0 holds). Thus, for each i we have a state where $\mathcal{L}(x_i)$ is true and a state where $\mathcal{L}(\neg x_i)$ is true. For every clause we create a state and for each literal l_j in the clause we add a transition to the state where $\mathcal{L}(l_j)$ is true. Therefore, for each clause c_i we have a state where $\mathcal{C}(c_i)$ is true. Finally, we add a state and transitions to all the states where some $\mathcal{C}(c_i)$ is true. Clearly, $\mathcal{F}(\phi_p)$ is true at that state, which we call the root state. Observe that $\mathcal{CA}(n)$ will also be satisfied in the root state independent of where y is true, because for every $i \in \{1, \ldots, n\}$ we have made a unique state p_i where v_i is true and p_i is at distance exactly 3 from the root.

Take a satisfying assignment; for every x_i which is true set the variable y to *true* at the state of the Kripke structure where v_i is true. Set y to *false* in every other state. Now, we must show that $\mathcal{S}$ is true at the root state. This is not hard to verify because for every clause in the original formula there is a true literal, call it l. If that literal is not negated then in the state where $\mathcal{L}(l)$ is true we have $\neg\Diamond v_0$ (because the literal is not negated) and $\Box y$ (because the literal is true, so its variable is true thus we must have set y to *true* at the variable's corresponding state). Therefore $(\neg\Diamond v_0 \rightarrow \Box y) \wedge (\Diamond v_0 \rightarrow \Box\neg y)$ is true at the literal's corresponding state and $\Diamond\left[(\neg\Diamond v_0 \rightarrow \Box y) \wedge (\Diamond v_0 \rightarrow \Box\neg y)\right]$

is true at the clause's corresponding state. Similar arguments can be made for a negated literal. Since we start with a satisfying assignment the same can be said for every clause, thus $\mathcal{S}$ is also true at the root state.

□

Now, we need to show that the produced modal formula has very small depth and the hardness result will follow in a way very similar to the results of [8].

Definition 2 $tow(h)$ is the inductively defined function $tow(0) = 0$ and $tow(h+1) = 2^{tow(h)}$.

Lemma 2 *Suppose that ϕ_p is a propositional CNF formula with n variables. Then, if $tow(h) \geq n$ the formula $\phi_m = \mathcal{F}(\phi_p) \wedge \mathcal{S} \wedge \mathcal{CA}(n)$ has modality depth at most $4 + h$.*

Proof First observe that the modality depth of ϕ_m is at most

$$3 + \max_{0 \leq i \leq n} \mathrm{md}(v_i).$$

Therefore, we just have to bound the modality depth of v_i.

We will use induction on h to show that $tow(h) \geq n \Rightarrow \mathrm{md}(v_n) \leq h+1$. For $h = 0$ we have $tow(h) \geq n \Rightarrow n = 0$, therefore $\mathrm{md}(v_0) = 1$ and the proposition holds.

Suppose that the proposition holds for h.

Observe that $\mathrm{md}(v_n) \leq 1 + \max_{0 \leq i \leq \log n}\{\mathrm{md}(v_i)\}$ because writing n in binary takes at most $\log n + 1$ bits. If we have $n \leq tow(h+1)$ then $\log n \leq tow(h)$. From the inductive hypothesis $\mathrm{md}(v_i) \leq h+1$ for $i \leq \log n$. Therefore, $\mathrm{md}(v_n) \leq h+2$ and the proposition holds.

□

Theorem 2 *There is no algorithm which can solve modal satisfiability in* K *for formulae with a single variable and modality depth d in time $f(d) \cdot poly(|\phi|)$ with $f(d) = O(tow(d-5))$, unless P=NP.*

Proof Suppose that there exists an algorithm A which in time $f(d) \cdot poly(|\phi|)$ can decide if a modal formula ϕ with modality depth d and just one variable is satisfiable. We will use this algorithm to solve propositional satisfiability in polynomial time.

Given a propositional CNF formula ϕ_p we construct ϕ_m as described, and if ϕ_p has n variables let $H = \min\{h \mid n \leq tow(h)\}$. Then $\mathrm{md}(\phi_m) \leq H + 4$ and of course ϕ_m can be constructed in time polynomial in $|\phi_p|$. Now we can use the hypothetical algorithm to see if ϕ_m is satisfiable.

We have that $f(d) = O(tow(d-5))$. Therefore, running this algorithm will take time $f(H+4) \cdot poly(|\phi_m|) = O(tow(H-1) \cdot poly(|\phi_m|))$. But by the definition of H we have $tow(H-1) \leq n$, therefore this bound is polynomial in $|\phi_m|$ and therefore, also in $|\phi_p|$, which means that we can solve an NP-complete problem in polynomial time.

□

4 Diamond Dimension

In this Section we propose a structural characteristic of modal formulae called diamond dimension. This is an alternative natural formula complexity measure which intuitively bounds the size of a model required to satisfy a formula. As we will see the parameter dependence of a satisfiability algorithm for formulae of small diamond dimension is doubly exponential, immensely lower than the dependence for modal depth. However, we will also show a lower bound indicating that it is unlikely that an algorithm with singly exponential parameter dependence could exist for this measure.

Definition 3 Let ϕ be a modal formula in negation normal form, that is, with the $\neg$ symbol appearing only directly before propositional variables. Then its diamond dimension, denoted by $\mathrm{d}_\Diamond(\phi)$ is defined inductively as follows:

- $\mathrm{d}_\Diamond(p) = \mathrm{d}_\Diamond(\neg p) = 0$, if p is a propositional variable
- $\mathrm{d}_\Diamond(\phi_1 \wedge \phi_2) = \mathrm{d}_\Diamond(\phi_1) + \mathrm{d}_\Diamond(\phi_2)$
- $\mathrm{d}_\Diamond(\phi_1 \vee \phi_2) = \max\{\mathrm{d}_\Diamond(\phi_1), \mathrm{d}_\Diamond(\phi_2)\}$
- $\mathrm{d}_\Diamond(\Box\phi) = \mathrm{d}_\Diamond(\phi)$
- $\mathrm{d}_\Diamond(\Diamond\phi) = 1 + \mathrm{d}_\Diamond(\phi)$

Our goal with this measure is to prove that if $\mathrm{d}_\Diamond(\phi)$ is small then ϕ's satisfiability can be checked in models with few states. This is why the two properties of ϕ which can increase $\mathrm{d}_\Diamond(\phi)$ are $\Diamond$ (which requires the creation of a new state) and $\wedge$ (which requires the creation of states for both parts of the conjunction).

Lemma 3 *If a modal formula ϕ is satisfiable and $\mathrm{d}_\Diamond(\phi) \leq k$ then there exists a Kripke structure with $O(2^{k/2})$ states which satisfies ϕ.*

Proof Suppose that there exists a Kripke structure which satisfies ϕ, that is there exists some state s in that structure where ϕ holds. We will construct a working set of modal formulae S which will satisfy the following properties:

(i) All formulae in S hold in s.
(ii) $(\bigwedge_{\phi_i \in S} \phi_i) \rightarrow \phi$ is a valid formula.
(iii) $\mathrm{d}_\Diamond(\phi) \geq \sum_{\phi_i \in S} \mathrm{d}_\Diamond(\phi_i)$.

We begin with $S = \{\phi\}$ which obviously satisfies the above properties. We will apply a series of transformations to S while retaining these properties until eventually we reach a point where every formula in S is simple (in a sense we will make precise later) and then we will construct a model with the promised number of states for ϕ.

While possible we apply the following rules to S:

1. If there exists a formula $\psi \in S$ such that $\psi = \psi^1 \wedge \psi^2$ then remove ψ from S and add ψ^1 and ψ^2 to S.
2. If there exists a formula $\psi \in S$ such that $\psi = \psi^1 \vee \psi^2$ then remove ψ from S. If ψ^1 is true at state s add ψ^1 to S, otherwise add ψ^2 to S.

3. If there are two formulae $\Box\psi_i$ and $\Box\psi_j$ in S then remove them and insert the formula $\Box(\psi_i \wedge \psi_j)$.

It should be clear that rule 1 does maintain the properties of S. Rule 2 also maintains the properties: property (i) is maintained because we assumed that ψ is true at state S therefore if ψ^1 is not true we add ψ^2 which must be true. The other properties are also straightforward. Finally, the third rule maintains the properties of S because of the fact that $\Box\psi_i \wedge \Box\psi_j \leftrightarrow \Box(\psi_i \wedge \psi_j)$ is a valid formula.

It should be clear that applying all the rules until none applies will take polynomial time. When we can no longer apply the rules we have that $S = \{\Box\psi, \Diamond\phi_1, \dots, \Diamond\phi_k, l_1, \dots, l_m\}$, where the l_i are propositional literals; in other words, we have (at most) one formula that starts with a $\Box$, say $\Box\psi$, and some number k of formulae that start with $\Diamond$, say $\Diamond\phi_i,\ 1 \leq i \leq k$.

Now we will use induction on the diamond dimension to prove the lemma. Let $s(d)$ be a function which upper bounds the number of states in the smallest model which are needed to satisfy formulae of diamond dimension d (we are going to calculate $s(d)$ recursively and prove that it is finite). First, we can say that $s(0) = 1$, because a formula with diamond dimension 0 has no diamonds. Therefore, S contains one formula that starts with a $\Box$ and some literals, for which there exists an assignment to make them all true (because of the first property of S). Clearly, a model with just one state where we pick this assignment will also make the formula that starts with $\Box$ trivially true, and by the second property of S will satisfy ϕ.

For the inductive step, suppose that all the satisfiable formulae of dimension at most $d = \mathrm{d}_\Diamond(\phi)$ need at most $s(d)$ states to be satisfied. Let's consider the diamond dimension of all the formulae in S. There are three cases: either S does not have a formula that starts with a $\Box$, or it doesn't have any formulae that start with $\Diamond$, or it has both.

If we have no formulae starting with diamonds we can easily see that the same model as in the base case suffices, since $\Box\psi$ is trivially true at a state without successors. So in this case we have just one state.

Suppose that all the formulae in S are literals or start with $\Diamond$. In this case, we have for all ϕ_i that $\mathrm{d}_\Diamond(\phi_i) \leq \mathrm{d}_\Diamond(\phi) - k$. Using the inductive hypothesis we get that the number of states to satisfy each formula ϕ_i is at most $s(\mathrm{d}_\Diamond(\phi_i))$. Clearly, we can create a model which is the union of the models for all the ϕ_i plus one state where we give an appropriate assignment to the literals and appropriate transitions so that $\Diamond\phi_i$ is true for all i. This model has at most $1 + \sum_{i=1}^{k} s(\mathrm{d}_\Diamond(\phi_i)) \leq 1 + k \cdot s(\mathrm{d}_\Diamond(\phi) - k)$ states.

Finally, if we have both types of formulae in S we construct the following model: consider all the formulae $\psi \wedge \phi_i$, for all i. Clearly, they are satisfiable, because $\Box\psi \wedge \Diamond\phi_i$ is true at s. We know from the third property of S that $\mathrm{d}_\Diamond(\phi) \geq \mathrm{d}_\Diamond(\psi) + k + \sum_{i=1}^{k} \mathrm{d}_\Diamond(\phi_i)$. Therefore, $\mathrm{d}_\Diamond(\psi \wedge \phi_i) = \mathrm{d}_\Diamond(\psi) + \mathrm{d}_\Diamond(\phi_i) \leq \mathrm{d}_\Diamond(\phi) - k - \sum_{j \neq i} \mathrm{d}_\Diamond \phi_j \leq \mathrm{d}_\Diamond(\phi) - k$. Now, we take the union of the models for each $\psi \wedge \phi_i$, and each model has at most $s(d - k)$ states. We add one state and transitions to the appropriate states where $\psi \wedge \phi_i$ are true, which together with an appropriate assignment makes all formulae of S true at that state. The number of states is at most $1 + k \cdot s(\mathrm{d}_\Diamond(\phi) - k)$.

From all the above cases we can upper bound $s(d)$ as $s(d) \leq \max_{1 \leq k \leq d}\{1 + k \cdot s(d-k)\}$. The fastest growing of these functions, obtained for $k = 2$, is in turn upper-bounded by $O(2^{d/2})$.

□

Theorem 3 *Given a modal formula ϕ with v variables and diamond dimension $\mathrm{d}_\Diamond(\phi) = k$ we can solve the satisfiability problem for ϕ in time $2^{O(2^k \cdot v)} \cdot |\phi|$.*

Proof From Lemma 3 it follows that if ϕ is satisfiable, this can be verified in a model of $O(2^{k/2})$ states. There are at most $2^{O(2^k)}$ Kripke frames from which we can get such models. For each we just enumerate through all possible assignments to the v variables in the $O(2^{k/2})$ states, a total of $2^{O(2^{k/2} \cdot v)}$ different assignments. Once we have fixed a model deciding if ϕ holds can be done in bilinear time. □

Lower Bound

We will now present a lower bound argument showing that, under reasonable complexity assumptions, the results we have shown for diamond dimension cannot be improved significantly. We will once again encode a propositional 3-CNF formula ϕ_p into a modal formula ϕ_m, this time with a goal of achieving small diamond dimension. We will also use a small number of propositional variables. We assume without loss of generality that we are given a 3-CNF formula ϕ_p with n variables, where n is a power of 2.

Let $\Box^j$ be short-hand for j consecutive repetitions of $\Box$, with $\Box^0\phi$ being equivalent to ϕ. We recursively define the formulae $F(i)$ as $F(0) := \Box\bot$ and $F(i) := \left(\Diamond(\bigwedge_{j=0}^{i-1} \Box^j b_i)\right) \wedge \left(\Diamond(\bigwedge_{j=0}^{i-1} \Box^j \neg b_i)\right) \wedge \Box F(i-1)$, where b_i are propositional variables. It is not hard to see that $\mathrm{d}_\Diamond(F(i)) = 2i$ and also that $F(i)$ can only be satisfied in a model with at least 2^i states. The model to keep in mind here is a complete binary tree of height i.

We will use the formula $F(\log n)$ to encode a 3-CNF formula with n variables and each leaf of the tree that must be constructed to satisfy it will correspond to a variable. It is now natural to encode the variables of the original formula using their binary representation. We define $B(x_k) := \bigwedge_{k_i=1} b_i \wedge \bigwedge_{k_i=0} \neg b_i$, where once again k_i denotes the i-th bit in the binary representation of k, now with the least significant bit numbered 1.

Our modal formula will also have a propositional variable y which will be true at leaves that correspond to variables of the 3-CNF formula that must be set to true. We encode a literal consisting of the variable x_k as $L_1(x_k) := \Box^{\log n}(B(x_k) \rightarrow y)$. The corresponding negated literal is $L_2(\neg x_k) := \Box^{\log n}(B(x_k) \rightarrow \neg y)$. A clause is encoded as the disjunction of the encodings of its three literals. Our final modal formula ϕ_m is a conjunction of $F(\log n)$ with the encodings of all the clauses of the propositional formula ϕ_p.

Lemma 4 *Given a propositional 3-CNF formula ϕ_p the modal formula ϕ_m is satisfiable in K iff ϕ_p is satisfiable.*

Proof Suppose that ϕ_p is satisfiable. We construct a binary tree of height $\log n$ as our model and ϕ_m will be made true at the root. It is not hard to satisfy $F(\log n)$ at the root: simply set $b_{\log n}$ to be true on all states on one of the subtrees of height $\log n - 1$ and false in all states of the other, then proceed to satisfy $F(\log n - 1)$ at the subtrees recursively in the same manner. Every leaf of the model corresponds to a variable of ϕ_p if we read the variables b_i as encoding the binary representation of the index of the variable. We set y to be true at the leaves that correspond to variables which are true at a satisfying assignment. It is not hard to see that this satisfies the encoding of all the clauses on the assumption that we started with an assignment satisfying ϕ_p.

Now for the other direction, suppose that ϕ_m is satisfied in a state of some model. A first observation is that for all $i \in \{1, \dots, n\}$ there must exist a state in which $B(i)$ holds and is at distance $\log n$ from the state where ϕ_m holds, as this is required for $F(\log n)$ to hold. From this we can infer that $\Box^{\log n}(B(i) \to y)$ and $\Box^{\log n}(B(i) \to \neg y)$ cannot both hold in the state where ϕ_m holds. Therefore, we can extract a consistent assignment for the variables of ϕ from the model, by setting to true the x_i for which $\Box^{\log n}(B(i) \to y)$ holds. It is not hard to see that this assignment must satisfy ϕ_p because its clauses are encoded in ϕ_m.

□

Now that we have described how to embed a 3-CNF formula into a modal formula with only logarithmically many variables and logarithmic diamond dimension we can use this fact to prove a lower bound. This time we rely on the stronger, but widely believed, assumption that 3-CNF SAT with n variables cannot be solved in time $2^{o(n)}$, also known as the Exponential Time Hypothesis (this is a standard assumption, see for example [16]). This allows us to obtain a much sharper bound than simply assuming that P≠NP.

Theorem 4 *There is no algorithm which can decide the satisfiability in K of a modal formula ϕ with v variables and $\mathrm{d}_{\Diamond}(\phi) = k$ in time $2^{2^{o(v+k)}} poly(|\phi|)$ unless the Exponential Time Hypothesis (ETH) fails.*

Proof Suppose that an algorithm running in time $2^{2^{o(v+k)}} poly(|\phi|)$ did exist. Then we could use the described construction to decide 3-CNF satisfiability for any formula with n variables. It is not hard to see that $v + k = O(\log n)$ and that the size of the produced modal formula is polynomial in the size of the 3-CNF formula, thus this would give an algorithm running in time $2^{o(n)}$, contradicting the ETH. □

5 Modal Width

In this section we give another structural parameter for modal formulae called modal width. We will show that satisfiability can be solved in time only singly exponential in the modal width and v. Thus, we will give an algorithm that works more efficiently for the class of modal formulae which have small width.

To give some intuition, the modal width measures how many different modal subformulae our formula contains at depth i. The idea is that the truth value of the subformulae of depth i at some state s depends only on the truth value of the subformulae of depth $i+1$ at the successors of s. If the maximum width of the formula is bounded we can exhaustively check all possible truth values for subformulae at the next level of depth and decide if some particular truth assignment to the subformulae of depth i is possible. Using this idea it is possible to obtain an algorithm with the promised running time if we use a dynamic programming technique.

First we define inductively the function $\mathrm{sub}(\phi)$ which given a modal formula returns a set of modal formulae. Intuitively, whether ϕ holds in a given state s of a Kripke structure depends on two things: the values of the propositional variables in s and the truth values of some formulae ψ_i in the successor states of s. These formulae are informally the subformulae of ϕ which appear at modal depth 1 and $\mathrm{sub}(\phi)$ gives us exactly this set of formulae.

- $\mathrm{sub}(p) = \emptyset$ if p is a propositional variable
- $\mathrm{sub}(\neg\phi) = \mathrm{sub}(\phi)$, $\mathrm{sub}(\phi_1 \vee \phi_2) = \mathrm{sub}(\phi_1 \wedge \phi_2) = \mathrm{sub}(\phi_1) \cup \mathrm{sub}(\phi_2)$
- $\mathrm{sub}(\Box\psi) = \mathrm{sub}(\Diamond\psi) = \{\psi\}$

Now we inductively define the set $S_i(\phi)$, which intuitively corresponds to the set of subformulae of ϕ at depth i.

- $S_1(\phi) = \mathrm{sub}(\phi)$
- $S_{i+1}(\phi) = \bigcup_{\psi \in S_i(\phi)} \mathrm{sub}(\psi)$

Finally, we can now define the modal width of a formula ϕ at depth i as $\mathrm{mw}_i(\phi) = |S_i(\phi)|$ and the modal width of a formula as $\mathrm{mw}(\phi) = \max_i \mathrm{mw}_i(\phi)$.

Observe that, as in the case of modal depth, negations do not affect the width of a formula. Therefore, the following results, which we state in terms of the satisfiability problem, also apply to the validity problem.

The following lemma is a basic observation regarding $\mathrm{mw}_i(\phi)$ and $\mathrm{md}(\phi)$.

Lemma 5 *For all $i \geq md(\phi)$ we have $mw_i(\phi) = 0$.*

Proof Observe that for all formulae ϕ such that $\mathrm{md}(\phi) \geq 1$ we have $\mathrm{md}(\phi) > \max_{\psi \in \mathrm{sub}(\phi)} \mathrm{md}(\psi)$. Using this fact the proof follows easily by induction on $\mathrm{md}(\phi)$.

□

Theorem 5 *There exists an algorithm which decides the satisfiability of a modal formula ϕ with v variables, $md(\phi) = d$ and $mw(\phi) = w$ in time $O(2^{2v+3w} \cdot d \cdot w \cdot |\phi|)$.*

Proof We will need to use a function *Prop*(ϕ) which, given a modal formula ϕ, returns a propositional formula which corresponds to ϕ with all modal subformulae replaced by new propositional variables. *Prop*(ϕ) can be inductively defined as follows (notice that once again we consider $\Diamond\phi$ as shorthand for $\neg\Box\neg\phi$):

- $Prop(p) = p$ if p is a propositional variable;
- $Prop(\phi_1 \vee \phi_2) = Prop(\phi_1) \vee Prop(\phi_2)$;
- $Prop(\phi_1 \wedge \phi_2) = Prop(\phi_1) \wedge Prop(\phi_2)$;
- $Prop(\neg\phi) = \neg Prop(\phi_1)$;
- $Prop(\Box\phi) = q_j$, where q_j is a new propositional variable.

Let $P = \{p_1, p_2, \dots, p_v\}$ be the set of propositional variables appearing in ϕ. For all $i \in \{0, \dots, d-1\}$, for all $P' \subseteq P$ and for all $S' \subseteq S_i(\phi)$ we define the formula $F(i, P', S')$,

$$F(i, P', S') = \bigwedge_{p_j \in P'} p_j \;\wedge \bigwedge_{p_j \in P \setminus P'} \neg p_j \;\wedge \bigwedge_{\psi \in S'} \psi \;\wedge \bigwedge_{\psi \in S_i(\phi) \setminus S'} \neg\psi.$$

Clearly there are at most $2^{v+w}d$ formulae $F(i, P', S')$ defined and for each one of these we will compute whether it is satisfiable or not using dynamic programming. We will use a boolean matrix $A(i, P', S')$ of size $2^{v+w}d$ to store the results.

First, we have $S_d(\phi) = \emptyset$. It is not hard to see that all formulae $F(d, P', \emptyset)$ are indeed satisfiable, so we initialize the corresponding entries in A to True. Suppose now that for some i we have filled out completely all entries $A(i + 1, P', S')$. We will show how to fill out any position in row i, say position $A(i, P', S')$. The crucial part now is that if we consider the formula $Prop(F(i, P', S'))$, it will have some new variables q_i which correspond to modal subformulae which all appear in $S_{i+1}(\phi)$.

The formula $Prop(F(i, P', S'))$ has at most $v+w$ variables. It is not hard to see that if $F(i, P', S')$ is satisfiable, then $Prop(F(i, P', S'))$ is also satisfiable, so our first step is to check this. The truth assignments for the v variables are easy to infer, therefore we only need to go through the 2^w possible assignments for the new variables. For each satisfying assignment we find we then need to check if a model that satisfies $F(i, P', S')$ can be built from it.

So, suppose that Q is the set of new variables, and we have found an assignment which sets the variables of $Q' \subseteq Q$ to *true* and the rest to *false* and satisfies $Prop(F(i, P', S'))$. Each variable q_j of Q corresponds to a formula $\Box\phi_j$ with $\phi_j \in S_{i+1}(\phi) \cup P$. If $q_j \in Q'$ we must make sure that ϕ_j is true at all successors of the state s where $F(i, P', S')$ will hold, in the model we are building. Let $S'' \subseteq S_{i+1}(\phi) \cup P$ be the set of formulae ϕ_j which we conclude that must hold in all successors of s in this way.

If $q_j \notin Q'$ we have that $\neg\Box\phi_j$ must hold in s, thus s must have a successor where $\neg\phi_j$ is true, or equivalently ϕ_j is false. Let $S^* \subseteq S_{i+1}(\phi) \cup P$ be the set of formulae ϕ_j for which we conclude that they must be false in some successor of s in this way.

To decide if it is possible to build appropriate successors to s so that all these conditions are satisfied, we look at row $i+1$ of A. Specifically we consider the set of entries $A(i+1, P', S')$ such that $S'' \subseteq S' \cup P'$ and $A(i+1, P', S') = T$. Informally, these correspond to formulae which are satisfiable (because the corresponding entry is set to *true*) and which also can serve as successors to s without violating the conditions of S'', that is, in any state where they hold all formulae which we need to be true at all successors of s are indeed true. Now, we simply check if for each $\phi_j \in S^*$ there exists an entry in the set we have selected so far with $\phi_j \notin S' \cup P'$. If this is the case we can conclude that $F(i, P', Q')$ is satisfiable and set the corresponding entry of A to *true*, otherwise we conclude that no satisfying model can be built from the assignment we get from Q, even though $Prop(F(i, P', S'))$ is satisfied. This whole process of computing S'' and S^* and checking through row $i+1$ of A can be performed in time $O(w \cdot 2^{v+w}|\phi|)$.

To decide if the initial formula ϕ is satisfiable, we compute $Prop(\phi)$ and perform the same process: for every satisfying assignment of $Prop(\phi)$ we look at corresponding entries of row 0 of A to see if a model for ϕ can be built. The total time for this algorithm is $O(2^{3w+2v}wd|\phi|)$, because for each of the at most $2^{v+w}d$ entries of A we need to check through at most 2^w assignments and for each we spend at most $O(w \cdot 2^{v+w}|\phi|)$.

□

Lower Bound

Intuitively, one would probably not expect that a significantly better algorithm is possible in this case, since the algorithm we have described is singly exponential in the parameter $v + w$. Indeed, it follows if one accepts the ETH that for formulae of width 0 (that is, propositional formulae) it is not possible to achieve time $2^{o(v+w)}$. Nevertheless, this kind of lower bound argument is not entirely satisfactory for our purposes, since it completely neglects the contribution of the modal width to the problem's hardness. For all we know, the best algorithm's dependence on w alone might be sub-exponential, though this would be surprising.

However, a more careful examination of the lower bound arguments we have presented for diamond dimension is useful here. The formulae constructed there have a logarithmic number of variables and linear modal width. Therefore, an algorithm which in general runs in time $2^{O(v)+o(w)}$ would in this case give an algorithm running in time $2^{o(n)}$ for propositional SAT, contradicting the ETH. In addition, even if one assumes a constant v, things cannot improve much. A second reading of the lower bound argument for modal depth shows that our construction has modal width $O(n \cdot polylog(n))$. This implies that any algorithm which runs in $2^{O(w^c)}$ for any $c < 1$ in the case of constant v would imply a $2^{o(n)}$ algorithm for SAT, again contradicting the ETH. Thus, the existence of an algorithm with significantly better dependence on w than the one presented here is unlikely.

6 Conclusions and Open Problems

In this paper we have defined and studied several modal formula complexity measures and investigated how each can be used to attack cases of modal satisfiability. Our results show that proving fixed-parameter tractability is only a first step in such problems, because the dependence on the parameters can vary significantly and some parameters offer much better algorithmic footholds than others.

It is worthy of remark that the measures of formula complexity we have discussed are not directly comparable; for example it is possible to construct a formula with small modality depth and very high modal width, and vice-versa. In this sense it is not possible to infer solely from our results which formula complexity measure is the "best", since each corresponds to a different family of modal formulae. However, our results can be seen as a first attempt at drawing a complexity "map" for different modal formula parameters, looking for areas where satisfiability becomes more or less tractable. This perspective creates a nice connection between this work and for example the research area of graph widths, where the complexity of model checking problems on graphs is explored in different graph families depending on a graph complexity measure. This is a well-developed area whose insights may be applicable and helpful in the study of the problems of this paper. (For a summary of the current complexity "map" for graph width parameters see Figure 8.1 in [9] - a more recent version of this paper appears in [6]).

A possible future direction is the investigation of yet more natural formula complexity measures. Additionally, extending our results to other modal logics, such as modal logics where Kripke structures are required to be reflexive or transitive (e.g. T, S4) would be an interesting next step. Another direction would be to extend this investigation to the multi-agent setting, where more formula complexity measures can be defined; it is known from the examples of logics S5 and KD45 that when making the transition from the single- to the multi agent setting, the picture in terms of computational complexity may change.

Acknowledgements: We would like to thank an anonymous reviewer for spotting an error in an earlier version of our proof of Theorem 2.

References

1. Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning).* Elsevier Science Inc., New York, NY, USA, 2006.
2. Alexander V. Chagrov and Mikhail N. Rybakov. How Many Variables Does One Need to Prove PSPACE-hardness of Modal Logics. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic*, pages 71–82. King's College Publications, 2002.
3. Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990.
4. R.G. Downey and MR Fellows. *Parameterized complexity.* Springer, 1999.
5. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* The MIT Press, 1995.
6. J. Flum, E. Grädel, and T. Wilke. *Logic and automata: history and perspectives.* Amsterdam Univ Pr, 2008.
7. J. Flum and M. Grohe. *Parameterized complexity theory.* Springer-Verlag New York Inc, 2006.

8. Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.
9. Martin Grohe. Logic, graphs, and algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(091), 2007.
10. Joseph Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artif. Intell.*, 75(2):361–372, 1995.
11. Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(3):319–379, 1992.
12. Joseph Y. Halpern and Leandro Chaves Rêgo. Characterizing the np-pspace gap in the satisfiability problem for modal logic. In Manuela M. Veloso, editor, *IJCAI*, pages 2306–2311, 2007.
13. Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.
14. L.A. Nguyen. On the complexity of fragments of modal logics. *Advances in Modal Logic*, 5:249–268, 2005.
15. E. Spaan. *Complexity of modal logics.* PhD thesis, University of Amsterdam, 1993.
16. G. Woeginger. Exact algorithms for NP-hard problems: A survey. *Combinatorial OptimizationEureka, You Shrink!*, pages 185–207.