Modal Logics with Hard Diamond-free Fragments

Antonis Achilleos

The Graduate Center of The City University of New York,
365 Fifth Avenue New York, NY 10016 USA
aachilleos@gc.cuny.edu

Abstract

We investigate the complexity of modal satisfiability for certain combinations of modal logics. In particular we examine four examples of multimodal logics and demonstrate that even if we restrict our inputs to diamond-free formulas, we still have a high complexity for the satisfiability problem for these logics. Our goal is to illustrate that having D as one or more of the combined logics, as well as the way logics are combined are important sources of complexity, even when in the absence of diamonds and even when at the same time we allow only one or two propositional variables.

1 Introduction

The complexity of the satisfiability problem for modal logic, and thus of its dual, modal provability/validity, has been extensively studied. Whether one is interested in areas of application of modal logic, or in the properties of modal logic itself, the complexity of modal satisfiability plays an important role. Ladner has established most of what are now considered classical results on the matter ([9]), determining that most of the usual modal logics, especially ones with more than one modalities are \texttt{PSPACE}-complete. Therefore, it makes sense to try to find fragments of these logics that have an easier satisfiability problem by restricting the modal elements of a formula. Much effort in this direction has been made in [7, 1, 10], examining how the complexity of this problem is affected when we restrict the number of propositional variables of a formula, its modal depth or width, its treewidth, or effectively the number of diamonds (negative boxes) that can be found in the formula. In this paper we present negative results for this direction and for certain cases of multimodal logics with connected modalities. For more on modal logic and its complexity, see [3, 5, 11].
A (mono)modal formula is a formula formed by using propositional variables and boolean connectives, much like propositional calculus, but we also use two additional operators, \( \Box \) (box) and \( \Diamond \) (diamond): if \( \phi \) is a formula, then \( \Box \phi \) and \( \Diamond \phi \) are formulas. Modal formulas are given truth values with respect to a Kripke model \((W, R, V)\), which can be seen as a directed graph \((W, R)\) together with a truth value assignment for the propositional variables for each world (vertex) in \(W\), called \(V\). \( \Box \phi \) is true in a world \(a\) if \(\phi\) is true at every \(b\) such that \((a, b)\) is an edge, while \(\Diamond \phi\) is the dual operator: \(\Diamond \phi\) is true at \(a\) if \(\phi\) is true at some \(b\) such that \((a, b)\) is an edge.

We are interested in the complexity of the satisfiability problem for modal formulas that have no diamonds - i.e. is there a model with a world at which our formula is true? When testing a modal formula for satisfiability (for example, trying to construct a model for the formula by using a tableau procedure), a clear source of complexity is the occurrence of diamonds in the formula. When we try to satisfy \(\Diamond \phi\), we need to assume the existence of an extra world where \(\phi\) is satisfied. Furthermore, when trying to satisfy \(\Diamond \phi_1 \land \Diamond \phi_2 \land \Box \phi_3\), we require two new worlds where \(\phi_1 \land \phi_3\) and \(\phi_2 \land \phi_3\) are respectively satisfied, which can potentially cause an exponential explosion to the size of the constructed model. There are several modal logics, but it is usually the case that in the process of satisfiability testing, as long as there are no diamonds in the formula, we are not required to add more than one world to the constructed model, which makes identifying the existence of diamonds as an important source of complexity a natural conclusion. On the other hand, when the modal logic is \(\mathbf{D}\), then its models are required to have a serial accessibility relation (no sinks in the graph). Thus, when we test \(\Box \phi\) for \(\mathbf{D}\)-satisfiability, we require a world where \(\phi\) is satisfied. In such a monomodal setting and in the absence of diamonds, we avoid an exponential explosion in the number of worlds and we can consider models with only a polynomial number of worlds.

Spaan in [11] and Demri in [4] have examined the complexity of combinations of modal logic. In particular, Demri studied \(L_1 \oplus \subseteq L_2\), which is \(L_1 \oplus L_2\) (see [11]) with the additional axiom \(\Box_2 \phi \rightarrow \Box_1 \phi\) and where \(L_1, L_2\) are among \(\mathbf{K}, \mathbf{T}, \mathbf{B}, \mathbf{S4}\), and \(\mathbf{S5}\). For when \(L_1\) is among \(\mathbf{K}, \mathbf{T}, \mathbf{B}\) and \(L_2\) among \(\mathbf{S4}, \mathbf{S5}\), he establishes EXP-hardness for \(L_1 \oplus L_2\)-satisfiability. In this paper we consider \(L_1 \oplus \subseteq L_2\), where \(L_1\) is a unimodal or bimodal logic (usually \(\mathbf{D}\), or \(\mathbf{D4}\)). When \(L_1\) is bimodal, \(L_1 \oplus \subseteq L_2\) is \(L_1 \oplus L_2\) with the extra axioms \(\Box_3 \phi \rightarrow \Box_1 \phi\) and \(\Box_3 \phi \rightarrow \Box_2 \phi\).

In this paper, we consider the effect on the complexity of modal satisfiability testing of restricting our input to diamond-free formulas under the requirement of seriality and in a multimodal setting with connected modalities. In particular we examine four examples: \(\mathbf{D}_2 \oplus \subseteq \mathbf{K}, \mathbf{D}_2 \oplus \subseteq \mathbf{K}_4, \mathbf{D} \oplus \subseteq \mathbf{K}_4, \) and \(\mathbf{D}_4 \oplus \subseteq \mathbf{K}_4\). For
these logics we look at their diamond-free fragment and establish that they are \( \text{PSPACE}\)-hard and in the case of \( D_2 \oplus \subseteq K4 \), \( \text{EXP}\)-hard. Furthermore, \( D_2 \oplus \subseteq K4 \) is \( \text{EXP}\)-hard even for their 1-variable fragments, while \( D \oplus \subseteq K4 \), and \( D_{42} \oplus \subseteq K4 \) are \( \text{PSPACE}\)-hard even for their 1-variable fragments.

Of course these results can be naturally extended to more modal logics, but we treat what we consider simple characteristic cases. For example, it is not hard to see that nothing changes when in the above multimodal logics we replace \( K \) by \( D \), or \( K4 \) by \( D4 \), as the extra axiom \( \Box_3 \phi \rightarrow \Diamond_3 \phi \) (\( \Box_2 \phi \rightarrow \Diamond_2 \phi \) for \( D \oplus \subseteq K4 \)) is a derived one. It is also the case that in these logics we can replace \( K4 \) by other logics with positive introspection (ex. \( S4, S5 \)) without changing much in our reasoning.

2 Modal Logics, Combinations, and Satisfiability

For the purposes of this paper it is convenient to consider modal formulas in negation normal form - negations are pushed to the propositional level and we have no implications - and this is the way we define our languages. We discuss modal logics with one, two, and three modalities, so we have three modal languages, \( L_1 \subseteq L_2 \subseteq L_3 \). They all include propositional variables, usually called \( p_1, p_2, \ldots \) (but this may vary based on convenience) and \( \bot \). If \( p \) is a propositional variable, then \( p \) and \( \neg p \) are called literals and are also included in the language and so is \( \neg \bot \), usually called \( \top \). If \( \phi, \psi \) are in one of these languages, so are \( \phi \lor \psi \) and \( \phi \land \psi \). Finally, if \( \phi \) is in \( L_3 \), then so are \( \Box_1 \phi, \Box_2 \phi, \Diamond_1 \phi, \Diamond_2 \phi, \Box_3 \phi, \Diamond_3 \phi \). In short, \( L_3 \) is defined in the following way:

\[
\phi ::= p \mid \neg p \mid \bot \mid \neg \bot \mid \phi \land \phi \mid \phi \lor \phi \mid \Diamond_1 \phi \mid \Box_1 \phi \mid \Box_2 \phi \mid \Box_3 \phi \mid \Diamond_2 \phi \mid \Diamond_3 \phi.
\]

\( L_2 \) includes all formulas in \( L_3 \) that have no \( \Box_3, \Diamond_3 \) and \( L_1 \) includes all formulas in \( L_2 \) that have no \( \Box_2, \Diamond_2 \). When we consider only formulas in \( L_1 \), \( \Box_1 \) will often just be called \( \Box \).

A Kripke model for a trimodal logic (a logic based on language \( L_3 \)) is a tuple \( \mathcal{M} = (W, R_1, R_2, R_3, V) \), where \( R_1, R_2, R_3 \subseteq W^2 \) and for every propositional variable \( p \), \( V(p) \subseteq W \). Then, \( (W, R_1, V) \) (resp. \( (W, R_1, R_2, V) \)) is a Kripke model for a monomodal (resp. bimodal) logic. \( (W, R_1), (W, R_1, R_2), \) and \( (W, R_1, R_2, R_3) \) are then called frames and \( R_1, R_2, R_3 \) are called accessibility relations. We define the truth relation \( \models \) between models, worlds (elements of \( W \), also called states) and formulas in the following recursive way:
\[
\mathcal{M} \models \neg \bot;
\]
\[
\mathcal{M}, a \models p \text{ iff } a \in V(p);
\]
\[
\mathcal{M}, a \models \neg \phi \text{ iff } \mathcal{M}, a \not\models \phi;
\]
\[
\mathcal{M}, a \models \phi \land \psi \text{ iff both } \mathcal{M}, a \models \phi \text{ and } \mathcal{M}, a \models \psi;
\]
\[
\mathcal{M}, a \models \phi \lor \psi \text{ iff either } \mathcal{M}, a \models \phi \text{ or } \mathcal{M}, a \models \psi;
\]
\[
\mathcal{M}, a \models \Box_i \phi \text{ iff there is some } b \in W \text{ such that } aR_ib \text{ and } \mathcal{M}, b \models \phi;
\]

finally, \[
\mathcal{M}, a \models \Diamond_i \phi \text{ iff for all } b \in W \text{ such that } aR_ib \text{ it is the case that } \mathcal{M}, b \models \phi.
\]

In this paper we deal with five logics: \(K, (D^2 \oplus_\subseteq K), (D^2 \oplus_\subseteq K4), (D \oplus_\subseteq K4), \) and \((D4^2 \oplus_\subseteq K4)\). All except for \(K\) and \((D \oplus_\subseteq K4)\) are trimodal logics, based on language \(L_3\). \(K\) is a monomodal logic (the simplest normal modal logic) based on \(L_1\), and \((D \oplus_\subseteq K4)\) is a bimodal logic based on \(L_2\). Each modal logic \(M\) is associated with a class of frames \(C\). A formula \(\phi\) is then called \(M\)-satisfiable iff there is a frame \(\mathcal{F} \in C\), a model \(\mathcal{M} = (\mathcal{F}, V)\), and a state \(a\) of \(\mathcal{M}\) such that \(\mathcal{M}, a \models \phi\). We then say that \(\mathcal{M}\) satisfies \(\phi\), or \(a\) satisfies \(\phi\) in \(\mathcal{M}\), or that \(\mathcal{M}\) models \(\phi\), or that \(\phi\) is true at \(a\).

\(K\) is associated with the class of all frames;

\((D^2 \oplus_\subseteq K)\) is associated with the class of frames \(\mathcal{F} = (W, R_1, R_2, R_3)\) for which \(R_1, R_2\) are serial (for every \(a\) there are \(b, c\) such that \(aR_1b, aR_2c\)) and \(R_1 \cup R_2 \subseteq R_3\);

\((D^2 \oplus_\subseteq K4)\) is associated with the class of frames \(\mathcal{F} = (W, R_1, R_2, R_3)\) for which \(R_1, R_2\) are serial, \(R_1 \cup R_2 \subseteq R_3\), and \(R_3\) is transitive;

\((D \oplus_\subseteq K4)\) is associated with the class of frames \(\mathcal{F} = (W, R_1, R_2)\) for which \(R_1\) is serial, \(R_1 \subseteq R_2\), and \(R_2\) is transitive;

\((D4^2 \oplus_\subseteq K4)\) is associated with the class of frames \(\mathcal{F} = (W, R_1, R_2, R_3)\) for which \(R_1, R_2\) are serial, \(R_1 \cup R_2 \subseteq R_3\) and \(R_1, R_2, R_3\) are transitive.
Tableau

A way to test for satisfiability is by using a tableau procedure. A good source on tableaux is \[3\]. We present tableau rules for \(K\) and for the diamond-free fragments of \((D^2 \oplus \subseteq) K\) and then for the remaining three logics. The reason we present these rules is because they are useful for later proofs and because they help give intuition regarding the way we can test for satisfiability. The ones for \(K\) are classical and follow right away. Formulas used in the tableau are given a prefix, which intuitively corresponds to a state in a model we attempt to construct. The tableau procedure for a formula \(\phi\) starts from 0 \(\phi\) and applies the rules it can to produce new formulas and add them to the set of formulas we construct, called a branch. A rule of the form \(\sigma \frac{a}{b} | c\) means that the procedure nondeterministically chooses between \(a\) and \(b\) to produce. If the branch has \(\sigma \bot\), or both \(\sigma p\) and \(\sigma \neg p\), then it is called propositionally closed and the procedure rejects its input. Otherwise, if the branch is closed under the rules and not propositionally closed, it is an accepting branch and the procedure accepts \(\phi\). The rules for \(K\) are:

\[
\begin{align*}
&\frac{\sigma \phi \lor \psi}{\sigma \phi} & \frac{\sigma \phi \land \psi}{\sigma \phi} & \frac{\sigma \square \phi}{\sigma \iota \phi} & \frac{\sigma \lozenge \phi}{\sigma \iota \phi} \\
&| (\sigma \psi) & \sigma \psi & & \\
& & & \text{for } \sigma \iota \text{ that has already appeared.} & \text{where } \sigma \iota \text{ new.}
\end{align*}
\]

For the remaining logics, we are only concerned with their diamond-free fragments, so we only present rules for those, since it makes things simpler. The rules for \((D^2 \oplus \subseteq) K\) follow:

\[
\begin{align*}
&\frac{\sigma \phi \lor \psi}{\sigma \phi} & \frac{\sigma \phi \land \psi}{\sigma \phi} & \frac{\sigma \square_1 \phi}{\sigma \iota_1 \phi} & \frac{\sigma \square_2 \phi}{\sigma \iota_2 \phi} & \frac{\sigma \square_3 \phi}{\sigma \iota_3 \phi} \\
&| (\sigma \psi) & \sigma \psi & \sigma \phi & \sigma \phi & \sigma \phi & \sigma \phi
\end{align*}
\]

We sketch a proof that these rules are correct. From an accepting branch we construct a model: let \(W\) be all the prefixes that have appeared in the branch, \(R_1 = \{(w, w.1) \in W^2\}, R_2 = \{(w, w.2) \in W^2\}, R_3 = \{(w, w.i) \in W^2| i \in \{1, 2\}\},\) and \(V(p) = \{w \in W| w p \text{ appears in the branch}\}.\) Then, it is not hard to see that \((W, R_1, R_2, R_3)\) is indeed a frame for \((D^2 \oplus \subseteq) K\), and that for \(M = (W, R_1, R_2, R_3, V), M, 0 \models \phi\) - by a straightforward induction on \(\phi\). Given some \(M, a \models \phi\), we can construct an accepting branch in the following way. We map 0 to \(a\) and for every \(w.i\), where \(i = 1, 2\) and \(w\) is mapped to state \(b\), then \(w.i\) is mapped to some state \(c\), where \(bR_{ri}c\). Then, we can easily make sure we make appropriate nondeterministic choices when applying a rule to ensure that if \(w \psi\)
is produced and \( w \) is mapped to \( a \), then always \( \mathcal{M}, a \models \phi \). Therefore, the branch can never be propositionally closed.

To come up with tableau rules for the other three logics, we can modify the above rules. The first two rules that cover the propositional cases are always the same, so we give the remaining rules for each case without proof. In the following, notice that the resulting branch may be infinite. However we can simulate such an infinite branch by a finite one: we can limit the size of the prefixes, as after a certain size it is guaranteed that there will be two prefixes that prefix the exact same set of formulas. Thus, we can either assume the procedure terminates or that it generates a full branch, depending on our needs.

The rules for the diamond-free fragment of \((D^2 \oplus \subseteq K4)\) are:

\[
\begin{align*}
\sigma \diamond 1 \phi & \quad \frac{\sigma \diamond 2 \phi}{\sigma.2 \phi} \\
\sigma.1 \phi & \quad \frac{\sigma.1 \phi}{\sigma.2 \phi}
\end{align*}
\]

The rules for the diamond-free fragment of \((D \oplus \subseteq K4)\) are:

\[
\begin{align*}
\sigma \diamond 1 \phi & \quad \frac{\sigma \diamond 2 \phi}{\sigma.1 \phi} \\
\sigma.1 \phi & \quad \frac{\sigma.2 \phi}{\sigma.1 \diamond 3 \phi}
\end{align*}
\]

The rules for the diamond-free fragment of \((D4^2 \oplus \subseteq K4)\) are:

\[
\begin{align*}
\sigma \diamond 1 \phi & \quad \frac{\sigma \diamond 2 \phi}{\sigma.1 \phi} \\
\sigma.1 \phi & \quad \frac{\sigma.2 \phi}{\sigma.1 \diamond 3 \phi}
\end{align*}
\]

We skip any proof for these cases, as they are similar to the previous case.
3 Lower Complexity Bounds for Diamond-free Fragments

In this section we give hardness results for the logics presented in the previous section - except for K. In [2], the authors prove that the variable-free fragment of K remains PSPACE-hard. We make use of that result here and prove the same for the diamond-free, 1-variable fragment of \((\mathbb{D}^2 \oplus_{\leq} \mathbb{K})\). Then we prove EXP-hardness for the diamond-free fragment of \(\mathbb{D}_2 \oplus_{\leq} \mathbb{K}_4\) and PSPACE-hardness for the diamond-free fragments of \(\mathbb{D} \oplus_{\leq} \mathbb{K}_4\) and of \(\mathbb{D}_{2\leq} \oplus_{\leq} \mathbb{K}_4\), which we later improve to the same result for the diamond-free, 1- or 2-variable fragments of these logics.

We now give a translation from unimodal formulas to formulas of three modalities such that \(\phi\) is \(\mathbb{K}\)-satisfiable if and only if \(\phi^{tr}\) (the result of the translation) is \((\mathbb{D}^2 \oplus_{\leq} \mathbb{K})\)-satisfiable. The translation is defined in the following way and we use an extra propositional variable (not appearing in \(\phi\)), \(q\).

For a formula \(\phi\), let \(\theta_1, \ldots, \theta_k\) be an enumeration of its subformulas viewed as distinct from each other and in increasing order with respect to their size. Also, let

\[
dseq : \{1, 2, \ldots, k\} \rightarrow \{\Box_1, \Box_2^\lceil \log(k+1) \rceil, \Diamond_3\}^{[\log(k+1)]}
\]

be some one-to-one mapping from those subformulas to a unique sequence of boxes. Then, we can recursively on \(i\) define \(i^{tr}\): If \(\theta_i = p\), (resp. \(\top\) or \(\bot\)) where \(p\) a propositional variable, then

- \(i^{tr} = p\) (resp. \(\top\) or \(\bot\));
- if \(\theta_i = \theta_j \circ \theta_l\), where \(\circ\) is either \(\land\) or \(\lor\), then \(i^{tr} = j^{tr} \circ l^{tr}\);
- if \(\theta_i = \neg \theta_j\), then \(i^{tr} = \neg j^{tr}\);
- if \(\theta_i = \Box \theta_j\), then \(i^{tr} = \Box_3^{[\log(k+1)]}(j^{tr} \lor \neg q)\);
- finally, if \(\theta_i = \Diamond \theta_j\), then \(i^{tr} = \Diamond_3(j^{tr} \land q)\).

Then, \(\phi^{tr} = k^{tr} \land q\) (as \(\theta_k\) is actually \(\phi\)).

It is not hard to see how each prefix \(\sigma\) such that \(\sigma \land q\) appears during the tableau procedure for \((\mathbb{D}_2 \oplus_{\leq} \mathbb{K})\) starting from \(\phi^{tr}\) corresponds to some prefix \(\sigma'\) that appears during the tableau procedure for \(\mathbb{K}\) starting from \(\phi\): map 0 to 0 and ensure that nondeterministic choices for one procedure are mirrored by choices for the other procedure; when a new prefix is generated by the procedure for \(\mathbb{K}\), because of the rule for \(\Diamond \theta_j\), then we can use the box rule repeatedly for \(i^{tr} = \Diamond_3(i)(j^{tr} \land q)\) and generate the corresponding prefix that is then mapped...
to that new prefix; on the other hand, if we generate a prefix $\sigma$ because of $i^{tr} = dseq(i)(j^{tr} \land q)$, we can mirror by using the diamond rule on $\theta_i$ for the other procedure and produce the prefix $\sigma$ maps to. When there is a prefix $\sigma$ that appears during the tableau procedure for $(\Delta_2 \oplus \subseteq K)$, but is not mapped to any prefix in the tableau for $K$, then we can just make the appropriate nondeterministic choice and produce $\sigma \neg q$. The extra variable, $q$, is needed to essentially mark the prefixes in the tableau that correspond to prefixes in the tableau for $K$ that have appeared.

Notice that $\chi^{tr}$ has no diamonds and the number of propositional variables in $\chi^{tr}$ is one more than in $\chi$. Since we can assume $\chi$ is variable-free ([2]), the following proposition follows.

**Proposition 1.** The diamond-free, 1-variable fragment of $(\Delta_2 \oplus \subseteq K)$ is PSPACE-hard.

We go on to prove hardness results for the remaining logics.

**Proposition 2.** The diamond-free fragment of $(\Delta_2 \oplus \subseteq K_4)$ is EXP-hard; the diamond-free fragments of $\Delta \oplus \subseteq K_4$ and of $\Delta_2 \oplus \subseteq K_4$ are PSPACE-hard.

**Proof.** We first treat the case of $(\Delta_2 \oplus \subseteq K_4)$. The proof is by reduction from an alternating Turing machine of two tapes (input and working tape) using polynomial space and resembles the one in [6]. Let the machine be $(Q, \Sigma, \delta, s)$, where $Q$ the set of states, $\Sigma$ the alphabet, $\delta$ the transition relation and $s$ the initial state. Let $Q = U \cup E$, where $E$ the set of existential and $U$ the set of universal states and assume that the machine only has two choices at every step of the computation, provided by two transition functions, $\delta_1, \delta_2$. Furthermore, let $x = x_1x_2 \cdots x_{|x|}$ be the input, where for every $i \in \{1, 2, \ldots, |x|\}$, $x_i \in \Sigma$. Since the Turing machine uses polynomial space, there is a polynomial $p$, such that the working tape only uses cells 1 to $p(|x|)$ for an input $x$. For the input tape, we only need cells 0 through $|x|+1$, because the head does not go any further and an output tape is not needed, since we are interested only in decision problems. Therefore, there are $Y, N \in Q$, the accepting and rejecting states respectively. Let $r_1 = \{0, 1, 2, \ldots, |x|+1\}$ and $r_2 = \{1, 2, \ldots, p(|x|)\}$.

For this reduction, a formula will be constructed that will enforce that any model satisfying it must describe a computation by the Turing machine. Each propositional variable will correspond to some fact about a configuration of the machine and the following propositional variables will be used:

- $t_1[i], t_2[j]$, for every $i \in r_1, j \in r_2$; $t_1[i]$ will correspond to the head for the first tape pointing at cell $i$ and similarly for $t_2[j]$,
• $\sigma_1[a, i], \sigma_2[a, j]$, for every $a \in \Sigma$, $i \in r_1, j \in r_2$; $\sigma_1[a, i]$ will correspond to cell $i$ in the first tape having the symbol $a$ and similarly for $\sigma_2[a, j]$ and the second tape,

• $q[a]$, for every $a \in Q$; $q[a]$ means the machine is currently in state $a$.

We need the following formulas. Intuitively, a state in a model for $\phi$ corresponds to a configuration of our Turing machine. $q$ ensures there is exactly one state at every configuration; $\sigma$ that there is exactly one symbol at every position of every tape; $t$ that for each tape the head is located at exactly one position; $\sigma'$ ensures that the only symbols that can change from one configuration to the next are the ones located in a position the head points at; $ac$ ensures we never reach a rejecting state (therefore the machine accepts); $st$ starts the computation at the starting configuration of the machine; finally, $d_E, d_U$ ensure for each configuration that the next one is given by the transition relation (functions). Then, if $com = q \land \sigma \land t \land \sigma' \land ac \land d_E \land d_U$,

$$\phi = st \land com \land \Box_2 com$$

$$q = \left( \bigvee_{a \in Q} q[a] \right) \land \bigwedge_{a, b \in Q, a \neq b} \neg (q[a] \land q[b]) ,$$

$$\sigma = \bigwedge_{j \in \{1, 2\}} \left( \bigvee_{a \in \Sigma} \sigma_j[a, i] \right) \land \bigwedge_{a, b \in \Sigma, a \neq b} \neg (\sigma_j[a] \land \sigma_j[b]) ,$$

$$t = \bigwedge_{j \in \{1, 2\}} \left( \bigvee_{i \in r_j} t_j[i] \right) \land \bigwedge_{i, k \in r_j, i \neq k} \neg (t_j[i] \land t_j[k]) ,$$

$$\sigma' = \bigwedge_{j \in \{1, 2\}, i, i' \in r_j, i \neq i', a \in \Sigma} \left[ (t_j[i] \land \sigma_j[a, i']) \rightarrow \Box_1 \sigma_j[a, i'] \land \Box_2 \sigma_j[a, i'] \right] ,$$

$$ac = \neg q[N] ,$$

$$st = \phi_{start} ,$$

where $\phi_{start}$ describes the initial configuration of the machine,
\[d_E = \bigwedge_{(a,i_1,i_2) \in E \times \Sigma \times \Sigma, \ j_1 \in r_1, \ j_2 \in r_2} \left[(q[a] \land \sigma_1[i_1,j_1] \land \sigma_2[i_2,j_2] \land t_1[j_1] \land t_2[j_2]\right) \]

\[\rightarrow \left[\Box_1 (q[a_1] \land \sigma_2[k_1,j_2] \land t_1[j_1 + m_1^1] \land t_2[j_2 + m_2^1]) \lor \Box_2 (q[a_2] \land \sigma_2[k_2,j_2] \land t_1[j_1 + m_1^2] \land t_2[j_2 + m_2^2])\right],\]

where \((a_1, k_1, m_1^1, m_2^1) = \delta_1(a, i_1, i_2), (a_2, k_2, m_1^2, m_2^2) = \delta_2(a, i_1, i_2)\).

\[d_U = \bigwedge_{(a,i_1,i_2) \in U \times \Sigma \times \Sigma, \ j_1 \in r_1, \ j_2 \in r_2} \left[(q[a] \land \sigma_1[i_1,j_1] \land \sigma_2[i_2,j_2] \land t_1[j_1] \land t_2[j_2]\right) \]

\[\rightarrow \left[\Box_1 (q[a_1] \land \sigma_2[k_1,j_2] \land t_1[j_1 + m_1^1] \land t_2[j_2 + m_2^1]) \land \Box_2 (q[a_2] \land \sigma_2[k_2,j_2] \land t_1[j_1 + m_1^2] \land t_2[j_2 + m_2^2])\right],\]

where \((a_1, k_1, m_1^1, m_2^1) = \delta_1(a, i_1, i_2), (a_2, k_2, m_1^2, m_2^2) = \delta_2(a, i_1, i_2)\).

The few implications that appear above are of the form \(a \land b \land \cdots \land c \rightarrow \psi\) (where \(a, b, \ldots, c\) are propositional variables) and can thus be rewritten in normal form: \(\neg a \lor \neg b \lor \cdots \lor \neg c \lor \psi\). For every configuration \(c\) of the Turing machine, there is a formula that describes it. This formula is the conjunction of the following and from now and on it will be denoted as \(\phi_c\): \(q[a]\), if \(a\) is the state of the machine in \(c\); \(t_1[i]\) and \(t_2[j]\), if the first tape’s head is on cell \(i\) and the second tape’s head is on cell \(j\); \(\sigma_1[a_1,i_1], \sigma_2[a_2,i_2]\), if \(i_1 \in r_1, i_2 \in r_2\) and \(a_1\) is the symbol currently in cell \(i_1\) of the first tape and \(a_2\) is the symbol currently in cell \(i_2\) of the second tape. Then, \(\phi_{\text{start}}\) is the same as \(\phi_{c_0}\), where \(c_0\) is the initial configuration for the machine on input \(x\).

Claim: If for some model \(M\), \(w \vDash \phi\) and for some \(u\), \(wR_3u\) and \(u \vDash \phi_c\) and \(c_1, c_2\) are the next configurations from \(c\), then if \(c\) a universal configuration, there are \(wR_3u_1, u_2\), such that \(u_1 \vDash \phi_{c_1}, u_2 \vDash \phi_{c_2}\) and if \(c\) an existential configuration, there is some \(wR_3u_1\), such that either \(u_1 \vDash \phi_{c_1}\) or \(u_1 \vDash \phi_{c_2}\). From this claim, it immediately follows that if \(\phi\) is satisfiable, then the Turing machine accepts its input. We prove the claim for the case of the universal configuration. Because of formulas \(q, \sigma, t\), in every state \(v\), such that \(wR_1v\), there is exactly one \(\phi_c\) satisfied. There are states \(u_1, u_2\), (because of seriality of \(R_2, R_3\)) such that \(wR_2u_1\) and \(wR_3u_2\) and if \(u_1 \vDash \phi_a, u_2 \vDash \phi_b\), then because of \(d_U\), \(a\) will differ from \(c\) in all respects \(\delta_1\) demands; furthermore, because of \(\sigma'\), \(a\) differs only in the ways \(\delta_1\) demands. Therefore, \(a = c_1\) (or perhaps \(c_2\), but this does not affect anything). Similarly, \(b = c_2\).

On the other hand, assuming that the Turing machine accepts \(x\), given its computation tree for \(x\), we can construct the following model \((W, R_1, R_2, R_3, V)\)
for $\phi$. $W$ is the set of configurations in the computation tree; let $R_1, R_2$ be minimal such that if $a$ is a universal configuration and $b,c$ its next configurations, then $aR_1b$ and $aR_2c$ (or $aRyb$ and $aR_1c$), while if $a$ an existential configuration and $b$ its next accepting configuration, then $aR_1b$ and $aR_2a$; let $R_3$ be the transitive closure of $R_1 \cup R_2$. $V$ is defined to be such that if $\mathcal{M} = (W, R_1, R_2, R_3, V)$, then $\mathcal{M}, a \models \phi_a$. Then, it is not hard to see that $\mathcal{M}, c_0 \models \phi$.

For the case of $D \oplus \mathbb{C}K4$, notice that if the machine is not alternating, but just (non)deterministic, we can eliminate $d_U$ and the subformulas beginning with $\Box_2$ from $\sigma'$ - of course, this means we rename the modalities from $\Box_1, \Box_2$ to $\Box_1, \Box_3$. For the case of $D4_2 \oplus \mathbb{C}K4$, we can define a translation from the language of $D \oplus \mathbb{C}K4$ to the language of $D4_2 \oplus \mathbb{C}K4$: given a formula $\phi$ with $\Box_1, \Box_2$ as modalities, $\phi^{tr}$ results from $\phi$ by replacing $\Box_2$ by $\Box_3$ and $\Box_1$ by $\Box_1 \Box_2$. The remaining argument is similar for the one for the case of $D2 \oplus \mathbb{C}K$ - the iteration of $\Box_1$ and $\Box_2$ helps cut off the propagation of boxes in the tableau, which does not happen for $D \oplus \mathbb{C}K4$.

We now present a method to translate a formula $\phi$ in negation normal form into a 1-variable formula $\phi'$ such that $\phi$ is $(D2) \oplus \mathbb{C}K4$-satisfiable iff $\phi'$ is $(D2) \oplus \mathbb{C}K4$-satisfiable. Let $p_1, \ldots, p_k$ be all the propositional variables that appear in $\phi$ and assume $q$ is not one of them. Then, $p_i^q = \Box_1 \Box_2 q$ and $(\neg p_i)^v = \Box_1 \Box_2 \neg q$. $\phi'$ results from $\phi$ by replacing each literal $l$ by $l^v$. Notice that in a model $\mathcal{M}$ and state $u$, only one of $p_i^u$ and $(\neg p_i)^u$ can be true. Let $\mathcal{M} = (W, R_1, R_2, R_3, V)$, where $(W, R_1 \cup R_2)$ is an infinite rooted tree, $u \in W$, the root, and $\mathcal{M}, u \models \phi$ (it is not hard to see how to construct such a model from any other). Then, for every $x \in W$, if there are some $y \in W$ and some positive $j \in \mathbb{N}$, such that $yR_1 R_2 x$ ($R_2^j$ is defined: $R_2^0 = R_2$ and $aR_2^{j+1}b$ iff there is some $c$ s.t. $aRa_2 R_2 b$, then $y, j$ are unique. Thus, if $V'(q) = \{x \in W \mid \exists yR_1 R_2 x \text{ s.t. } y \in V(p_i)\}$, it is the case that for $\mathcal{M'} = (W, R_1, R_2, R_3, V')$, $\mathcal{M'}, u \models \phi'$. On the other hand given a model $\mathcal{M'}, u \models \phi'$, we can just define $V(p_i) = \{x \in W \mid \mathcal{M'}, x \models \Box_1 \Box_2 q\}$, thus $\phi$ is satisfiable iff $\phi'$ is. If $\phi$ is diamond-free, then $\phi'$ remains diamond-free, so we can conclude with the following:

**Proposition 3.** The diamond-free, 1-variable fragment of $D_2 \oplus \mathbb{C}K4$ is EXP-hard.

Notice that the method above does not work for $D \oplus \mathbb{C}K4$. Thus we use another method: we translate a formula $\phi$ to a formula $\phi^2$ such that $\phi$ is $D \oplus \mathbb{C}K4$-satisfiable iff $\phi^2$ is $D \oplus \mathbb{C}K4$-satisfiable. Let $p_1, \ldots, p_k$ be the propositional variables that appear in $\phi$ and let $p, q$ be new variables (not among $p_1, \ldots, p_k$). Replace all instances of $\Box_1 \psi$ in $\phi$ by $\Box_1^{k+1}(\psi \land p)$ ($\Box_1^{k+1}$ is $k+1$ iterations of $\Box_1$), all instances of $\Box_2 \psi$ in $\phi$ by $\Box_2(\psi \lor \neg p)$, all instances of $p_i$ (without negation) by $\Box_1^i q$, and all instances of $\neg p_i$ by $\Box_1^i \neg q$. We can end this argument like the one for the case of $D_2 \oplus \mathbb{C}K$ and we conclude with the following.
Proposition 4. The diamond-free, 2-variable fragments of $D \oplus \subseteq K4$ and of $D4_2 \oplus \subseteq K4$ are PSPACE-hard.

Final Remarks

One may wonder whether we can say the same for the variable-free fragment of these logics. The answer however is that we cannot. The models for these logics have accessibility relations that are all serial. This means that any two models are bisimilar when we do not use any propositional variables, thus any satisfiable formula is satisfied everywhere in any model, thus we only need one prefix for our tableau and we can solve satisfiability recursively on $\phi$ in polynomial time.

Notice that for the proofs above, the requirement that the respective accessibility relations are serial was central. Indeed, otherwise there was no way to achieve these results, as we would not be able to force extra worlds in a constructed model. Then we would have to rely on the complexity contributed by propositional reasoning and at best we would get an NP-hardness result - as long as we allowed enough variables in our formula.

Whether we can extend these results to the 1-variable fragments of $D \oplus \subseteq K4$ and $K4_2 \oplus \subseteq K4$ is an open question.

References

[1] Antonis Achilleos, Michael Lampis, and Valia Mitsou. Parameterized modal satisfiability. *Algorithmica*, 64(1):38–55, 2012.
[2] Alexander V. Chagrov and Mikhail N. Rybakov. How many variables does one need to prove pspace-hardness of modal logics. In Advances in Modal Logic, pages 71–82, 2002.
[3] Marcello D’Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Posegga. Handbook of tableau methods. Springer, 1999.
[4] Stéphane Demri. Complexity of simple dependent bimodal logics. In Roy Dyckhoff, editor, *TABLEAUX*, volume 1847 of Lecture Notes in Computer Science, pages 190–204. Springer, 2000.
[5] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.
[6] Michael J Fischer and Richard E Ladner. Propositional dynamic logic of regular programs. *Journal of computer and system sciences*, 18(2):194–211, 1979.
[7] Joseph Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75:361–372, 1995.
[8] 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.

[9] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[10] M. Praveen. Does treewidth help in modal satisfiability? In Petr Hlinn and Antonn Kuera, editors, *Mathematical Foundations of Computer Science 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 580–591. Springer Berlin Heidelberg, 2010.

[11] E. Spaan. *Complexity of modal logics*. PhD thesis, University of Amsterdam, 1993.