QBF Reasoning for Concept Satisfiability via an Optimised Reduction From K

Momen Hassan[1], Uli Sattler[1]

Abstract

It is well known that satisfiability for the modal logic **K**, with multiple modalities, is a notational variant of concept satisfiability in the description logic $\mathcal{ALC}$. Likewise, extensions of K are notational variants of more expressive description logics. This close relationship extends to their reasoning tasks and is well understood. The same cannot be said for Quantified Boolean Formulas and Modal/Description Logic: despite their reasoning tasks belonging to the same complexity class, reasoners for QBF and **K** have been developed and investigated mostly separately. While there have been some attempts at using QBF solvers to solve modal logic formulae, such attempts were made before recent significant optimisations of QBF reasoners and have not resulted in a good understanding of the relationship between measures of difficulty in both logics, particularly the polynomial hierarchy division of classes of modal formulae. We define a translation from **K** formulae to closed QBF formulae that maps a satisfiable formula to a true one, with an optimisation to close the gap in understanding of how the polynomial hierarchy relates to modal logic - and consequently concept satisfiability in $\mathcal{ALC}$. We empirically evaluate our translation together with a modern QBF reasoner, depQBF, against native state-of-the-art modal logic reasoners. We show that the translation-based reasoner compares very favourably, in some cases vastly outperforming the native reasoners.

1. Introduction

The Modal Logic **K**, the Description Logic $\mathcal{ALC}$, and *Quantified Boolean Formulas* (QBF) are all extensions of propositional logic with PSPACE-complete satisfiability problems [1, 2, 3, 4]. Since **K** (or their multi-modal extension $\mathbf{K}_m$) formulae are a notational variant of $\mathcal{ALC}$ concepts [5], research into automated reasoning in **K** and its more expressive extensions has been closely tied to research in description logics, sharing their reasoning paradigms and theoretical understanding of the difficulty of formulae [6, 7, 8]. In contrast, there is little practical influence between QBF reasoning approaches and those for description logic.

Being a straightforward classical extension to propositional logic, QBF reasoners are able to leverage methods in reasoning from propositional logic. In particular, a powerful and heavily researched algorithm for satisfiability solving in propositional logic known as *Conflict Driven Clause Learning* or CDCL [9], has a simple counterpart in QBF, Quantified CDCL [10]; no such CDCL counterpart has been devised for modal or description logic to date.

The topic of the most effective algorithms for reasoning in modal logic has been a subject of debate since the inception of modal logic reasoners [11]. Variations of tableaux-based algorithms are common, but SAT-based algorithms, resolution-based algorithms and BDD-based algorithms have all been implemented without a clear consensus on which algorithms are most effective and for which kinds of formulae [8]. While tableaux remains the basis for understanding modal and description logics, more recently, *Counter Example Guided Abstraction Refinement*, CEGAR for short, has shown promise for reasoning in modal logic [12, 13]. CEGAR iteratively reduces modal logic formulae to abstracted propositional logic formulae, using an oracle propositional logic reasoner to solve these formulae, and refining the propositional logic abstraction if the oracle solver cannot find a solution. In this way, CEGAR utilises the power of propositional logic reasoners to solve modal logic formulae and has proven to solve modal logic benchmarks more effectively than 'native' modal logic reasoners [12].

DL 2026: 39th International Workshop on Description Logics, July 17–19, 2026, Lisbon, Portugal

Momen.Hassan@manchester.ac.uk (M. Hassan); Uli.Sattler@manchester.ac.uk (U. Sattler)
https://www.cs.man.ac.uk/~sattler/ (U. Sattler)
0009-0001-4669-4891 (M. Hassan); 0000-0003-4103-3389 (U. Sattler)

published 2026-07-25

Understanding of formulae difficulty is more advanced in QBF than in $\mathbf{K}$ and $\mathcal{ALC}$. The satisfiability problem of these logics exhibits both *non-deterministic difficulty* - the difficulty involved in selecting a correct choice out of an exponential number of options - and *co-non-deterministic* difficulty - the difficulty involved in constructing a possibly exponentially-sized model. It is the presence, and possibility of *alternating between* these two forms of difficulty that characterises the complexity class PSPACE, to which reasoning in these logics belongs. In fact, the class PSPACE is a union of an infinite set of complexity classes known as the *polynomial hierarchy*, with each class characterised by a bounded number of alternations between non-deterministic and co-non-deterministic difficulty. For QBF, alternations between universal and existential quantifiers in the prefix of formulae correspond to alternations between co-non-deterministic and non-deterministic difficulty respectively, and this quantity can be measured in linear time [3]. For modal and description logic, co-non-deterministic difficulty is measured by the nesting depth of modal (respectively role) operators which can be measured in linear time; however, the exact alternation between co-non-deterministic and non-deterministic difficulty in modal and description logic has so far not been demonstrated to be measurable in polynomial time. Not only is this discrepancy between QBF and modal/description logic theoretically interesting, it also means that understanding testcases for $\mathbf{K}$ reasoners and evaluating their performance poses significant problems [14, 15, 16]. Furthermore, the creation of testcases of precise difficulty relies on translations from QBF to $\mathbf{K}$ [17].

We present an optimised translation from $\mathbf{K}$ to QBF, returning a tighter upper bound on the alternation depth of modal formulae than the only other previously attempted translation [18]. Using this translation, we also conduct an empirical evaluation of a QCDCL QBF reasoner, *depQBF*, against the fastest performing $\mathbf{K}$ reasoner and a performant tableaux reasoner, showing that the QBF reasoner can, in some cases, far outperform the modal reasoners. Our work demonstrates the promise of drawing theoretical understanding of formulae and reasoning from QBF to modal and description logic.

2. Related Work

Other translation-based modal logic solvers have been attempted before, with varying degrees of success. Aside from recent CEGAR-based attempts, translations from modal to propositional logic have shown the most promise. Sebastiani et. al. used a complete translation from modal to propositional logic, with good performance on certain kinds of formula, especially those with high propositional difficulty and low modal difficulty [19]. Of course, given that satisfiability for propositional logic is NP-complete, translations from modal to propositional logic involve an exponential blow up in the size of formulae; given this, the success of a complete translation to propositional logic was limited to formulae with low modal depth. A smarter, incremental reduction to propositional logic achieved better results [20].

There has been one other translation of $\mathbf{K}$ satisfiability to QBF [18]. This translation did not consider the polynomial hierarchy and, in that regard, is equivalent to the unoptimised version of our translation in Section 4.4, with other differences in presentation and the form of translated formulae. To our knowledge, this translation has been evaluated with QBF reasoners two times, achieving quite poor results, falling behind native modal reasoners [18, 19]. However, both studies were made some time ago, using QBF reasoners which are now outdated. In particular, neither study used a QCDCL reasoner on their translated formulae. We present a new reduction from modal logic, enabling a polynomial hierarchy optimisation and use a more modern, QCDCL QBF solver, DepQBF, in our evaluation [21].

Regarding the polynomial hierarchy of modal formulae, Das et. al. attempted a measurement of the alternation depth of modal formulae [22]. In their work, they consider validity rather than satisfiability, though both reasoning tasks are PSPACE-complete. They do not claim to find an accurate, polynomial-time measure of alternation depth, nor do they provide a translation from modal formulae to QBF, rather they present a polynomial-time function which they claim returns an upper bound of the alternation depth of a modal formula and present a validity-preserving translation from QBF to modal logic for which their function returns the correct alternation depth of translated QBF. However, they do not provide a proof of their upper bound function and we believe there are classes of modal formulae

for which their function under-estimates the alternation depth, in which case their final result of a polynomial-time polynomial hierarchy division of modal formulae would be invalid.

3. Background

We assume the reader to be familiar with description logic or modal logic $\mathbf{K}$ [2, 1] and QBF [3], and will only briefly introduce some notation.

Propositional Logic Assuming a countably infinite set of propositional variables $P = \{p_1, p_2, p_3, ...\}$, we define the set of propositional logic formulae $\mathcal{F}$ as usual and over the operators $\wedge$, $\vee$, and $\neg$. We also use the implication $\varphi_1 \rightarrow \varphi_2$ as shortcut for $\neg\varphi_1 \vee \varphi_2$. Given a formula φ, we write $\mathsf{Vars}(\varphi) \subseteq P$ to denote the set of propositional variables occurring in φ. (Un)satisfiability, validity, and equivalence are defined as usual. A formula is in negation normal form, NNF, if negation is applied to propositional variables only. Using De Morgan's laws, we can transform each formula into an equivalent one of linear length in NNF.

Quantified Boolean Formulae We define the set of Quantified Boolean formulae in Prenex Normal Form (PNF), Q_{PNF}, together with a function returning the free variables of a formula in Q_{PNF}, $\mathsf{FVars}(\phi)$, recursively from the set of propositional logic formulae:

- base: if $\varphi \in \mathcal{F}$ and in NNF, then $\varphi \in Q_{\text{PNF}}$ and $\mathsf{FVars}(\varphi) = \mathsf{Vars}(\varphi)$
- exists: if $\varphi \in Q_{\text{PNF}}$ and $p \subseteq \mathsf{FVars}(\varphi)$, then $\exists p.\varphi \in Q_{\text{PNF}}$ and $\mathsf{FVars}(\exists p.\varphi) = \mathsf{FVars}(\varphi) \setminus \{p\}$
- forall: if $\varphi \in Q_{\text{PNF}}$ and $p \subseteq \mathsf{FVars}(\varphi)$, then $\forall p.\varphi \in Q_{\text{PNF}}$ and $\mathsf{FVars}(\forall p.\varphi) = \mathsf{FVars}(\varphi) \setminus \{p\}$

To simplify notation, we merge sequences of the same quantifier together and quantify over *sets* of variables so that, for example, we can write $\exists e_1 \exists e_2 \exists e_3 \forall a_1 \forall a_2.\phi$ as $\exists\{e_1, e_2, e_3\}\forall\{a_1, a_2\}.\phi$. We call the leftmost quantifier the outermost quantifier and the rightmost quantifier the innermost quantifier. We say that a formula $\varphi \in Q_{\text{PNF}}$ is closed iff $\mathsf{FVars}(\varphi) = \varnothing$. for a Q_{PNF} formula $\psi = P.\phi$, where $\phi \in \mathcal{F}$ and P is a sequence of quantifiers over sets of variables, we call P the prefix and ϕ the matrix of ψ.

The modal logic $\mathbf{K}$ The modal logic $\mathbf{K}$ is the logical core of $\mathcal{ALC}$, i.e., $\mathbf{K}$ formulae are simply a notational variant of $\mathcal{ALC}$ concepts with one role [5]. $\mathbf{K}$ extends propositional logic with the unary necessity - $\Box$ - and possibility - $\Diamond$ operators. These are the notational equivalents of the universal - $\forall$ - and existential - $\exists$ - restriction operators in $\mathcal{ALC}$, respectively, e.g., $\exists r.(\forall r.(p \sqcap \exists r.\neg p))$ is a notational variant of $\Diamond(\Box(p \wedge \Diamond\neg p))$. We consider only $\mathbf{K}$ with one modality (respectively $\mathcal{ALC}$ with one role) as its satisfiability problem is in the same complexity class as its counterpart with multiple modalities, $\mathbf{K}_m$, and benchmark formulae are more difficult with one modality. In the conclusion, we describe a simple extension of our translation to multiple modalities or roles.

We consider $\mathbf{K}$ formulae in NNF, referred to as $\mathbf{K}_{NNF}$ and refer to subformulae at depth i in φ as those sub-formulae nested in i box or diamond occurrences. Then $\mathsf{Depth}(\varphi)$ is the maximum depth at which sub-formulae are nested in φ; for example, $\mathsf{Depth}(\Diamond(\Box(\Diamond(p \wedge q)))) = 3$.

Satisfiability of $\mathbf{K}$-formulae is PSpace-complete and every satisfiable $\mathbf{K}$ formula φ has a finite, tree-shaped model of depth at most $\mathsf{Depth}(\varphi)$ [1, 2].

Semantics of $\mathbf{K}$ and Q_{PNF} We assume the reader to be familiar with tree models of $\mathbf{K}$. A model of a closed Q_{PNF} formula, ψ - also commonly referred to as a *winning strategy* in the literature - is a partial binary tree with each non-leaf node labelled by a variable quantified in the prefix of ψ; each node labelled by a universally quantified variable has two children, representing states in which that variable is assigned true and false, while each existentially quantified variable has just one child, being either true or false. A QBF model alternates between sets of existentially and universally quantified variables following the prefix order of ψ, with each breadth of nodes in the model labelled by the same variable in the corresponding set of the quantifier in ψ's prefix.

4. QBF Translations

We translate $\mathbf{K}$ formulae in NNF to closed QBF. We begin by describing the core idea behind the translation and setting up the groundwork needed for the translation. We then describe the translation in two parts: describing the prefix of the translated formula before describing the matrix of the translated formula, making reference to its prefix. Finally, we describe a more optimal prefix for a translated formula, and present an optimised backwards translation from QBF to $\mathbf{K}$ to demonstrate its optimality.

4.1. Core Translation

The translation maps a satisfiable $\mathbf{K}$ formula ϕ to a true closed QBF ϕ' by creating a correspondence between a tree model of the original $\mathbf{K}$ formula and a winning strategy of the translated QBF such that there exists a winning strategy of the translated QBF if and only if there exists a tree model satisfying the original $\mathbf{K}$ formula. Furthermore, a winning strategy of the translated QBF can be converted, in time polynomial in the size of the winning strategy, to a tree model satisfying the original $\mathbf{K}$ formula. We use ϕ for the $\mathbf{K}$ formula to be translated and ϕ' for its translation.

In the closed QBF translation ϕ' of ϕ, we use sets of universally quantified variables to enumerate the indices of nodes in a potential tree model of ϕ. There is one such set for each depth in ϕ. We refer to these universally quantified sets as USet_i for each i with $1 \leq i \leq \mathsf{Depth}(\phi)$. Each USet_i in ϕ' is used to index nodes at depth i in tree models of ϕ. A valuation - assignment - of the variables in USet_i specifies an index in tree models of ϕ at depth i, and thus a valuation of the sets USet_1, USet_2, ... , USet_i specifies a complete path to a single node in a tree model of ϕ at depth i. We associate a unique node in all possible tree models with each diamond sub-formula.

For example, consider $\phi := \Diamond a \wedge \Diamond b \wedge \Box(\Diamond c \wedge \Diamond d)$. Here $\mathsf{Depth}(\phi) = 2$. At depth 1, we give $\Diamond a$ a value 1 and $\Diamond b$ value 2. At depth 2, we give $\Diamond c$ value 1 and $\Diamond d$ value 2. Let $\mathsf{USet}_1 := \{u_2^1, u_1^1\}$ and $\mathsf{USet}_2 := \{u_2^2, u_1^2\}$. Then the corresponding valuations of the diamonds' respective nodes in any tree model in consideration over USet_1 and USet_2 are $\{u_2^1 \mapsto 0, u_1^1 \mapsto 1\}$ for $\Diamond a$, $\{u_2^1 \mapsto 1, u_1^1 \mapsto 0\}$ for $\Diamond b$, $\{u_2^2 \mapsto 0, u_1^2 \mapsto 1\}$ for $\Diamond c$ and $\{u_2^2 \mapsto 1, u_1^2 \mapsto 0\}$ for $\Diamond d$ i.e. the binary encodings of 1 and 2 over USet_1 and USet_2.

4.2. Translation Preliminaries

For a formula ϕ in $\mathbf{K}_{NNF}$, we call a sub-formula $\Diamond\phi'$ of ϕ a *diamond* of ϕ and a sub-formula $\Box\phi'$ of ϕ a *box* of ϕ. For $i \leq \mathsf{Depth}(\phi)$, we use

- $\mathsf{Dias}_i(\phi)$ for the set of all diamonds occurring at a depth i in ϕ,
- $\mathsf{Boxs}_i(\phi)$ for the set of all boxes occurring at depth i in ϕ, and
- $\mathsf{Vars}_i(\phi)$ for the set of all propositional variables occurring at depth i in ϕ.

For example, let

$$\phi := a \wedge \Diamond a \wedge (\Diamond b \vee \Diamond c \vee \Box(\Diamond b \wedge \Diamond d)).$$

Then $\mathsf{Dias}_0(\phi) = \{\Diamond a, \Diamond b, \Diamond c\}$, $\mathsf{Dias}_1(\phi) = \{\Diamond b, \Diamond d\}$, $\mathsf{Boxs}_0(\phi) = \{\Box(\Diamond b \wedge \Diamond d)\}$, $\mathsf{Boxs}_1(\phi) = \emptyset$, $\mathsf{Vars}_0(\phi) = \{a\}$, $\mathsf{Vars}_1(\phi) = \{a, b, c\}$, $\mathsf{Vars}_2(\phi) = \{b, d\}$.

Our translation into QBF operates on $\mathbf{K}$ formulae in negation normal form with a "uniqueness" assumption: without loss of generality (see, e.g. [23]), we can assume that each propositional variable occurs at at most one depth in our formulae. The transformation of a $\mathbf{K}_{NNF}$ formula φ into an equi-satisfiable one of the same length satisfying this uniqueness constraint $\mathsf{Dec}(\varphi, 0)$ can be achieved through a simple recursive function to decorate variables with the depth they occur at:

- If $\phi = \varphi_1 \circ \varphi_2$ for $\circ \in \{\wedge, \vee\}$, then $\mathsf{Dec}(\phi, n) = \mathsf{Dec}(\varphi_1, n) \circ \mathsf{Dec}(\varphi_2, n)$
- If $\phi = \neg\varphi$ then $\mathsf{Dec}(\phi, n) = \neg\mathsf{Dec}(\varphi, n)$
- If $\phi = \bigcirc(\varphi)$ for $\bigcirc \in \{\Diamond, \Box\}$ then $\mathsf{Dec}(\phi, n) = \bigcirc(\mathsf{Dec}(\phi, n+1))$
- If $\phi = p$ where p is a propositional variable then $\mathsf{Dec}(\phi, n) = p_n$

So, in the following we assume without loss of generality that our **K** formulae are in NNF and satisfy our uniquenss constraint.

4.3. Prefix

For modal formula ϕ, the prefix of its translated QBF ϕ' is:

$$\exists \mathsf{ESet}_0 \forall \mathsf{USet}_1 \exists \mathsf{ESet}_1 ... \forall \mathsf{USet}_d \exists \mathsf{ESet}_d \tag{1}$$

where $d = \mathsf{Depth}(\phi)$. For $0 < i \leq d$, we define USet_i as $\mathsf{USet}_i = \{u_1^i, u_2^i \ldots, u_n^i\}$ where $n = \lceil log_2([\mathsf{Dias}_{i-1}(\phi)] + 1)\rceil$ i.e. each universal set contains unique variables, and each contains enough variables to enumerate the set of diamonds occurring at its depth.

We define each existential set, ESet_i, for $0 \leq i < d$ as $\mathsf{ESet}_i = \{e_1^i, e_2^i, ..., e_n^i\} \cup \mathsf{Vars}_i(\phi)$ where $n = [\mathsf{Dias}_i(\phi)] + [\mathsf{Boxs}_i(\phi)]$ i.e. enough variables for each diamond and box at each depth to be assigned a unique existential variable. We then define $\mathsf{ESet}_d = \mathsf{Vars}_d(\phi) \cup \mathsf{Wit}^\phi$ where $\mathsf{Wit}^\phi = \{wit_1, wit_2, .., wit_d\}$ is a set of variables, one for each depth in ϕ, representing that a diamond at that depth has been branched into i.e. there is a 'witness' requiring boxes at that depth to be satisfied.

These sets are sized such that injective mappings from diamonds and boxes at depth i to variables in ESet_i, and from diamonds at depth $i-1$ to enumerations of USet_i, can be defined for a formula ϕ. We therefore assume injective mappings from the diamonds to sets of universal variables $\mathsf{UMap}_i : \mathsf{Dias}_i(\phi) \mapsto 2^{\mathsf{USet}_{i+1}}$ for $0 \leq i < \mathsf{Depth}(\phi)$; the set of universals a diamond is mapped to at depth $i-1$ represents the subset of USet_i which is set to true in an enumeration of that diamond. We also assume injective mappings from diamonds and boxes to existential variables $\mathsf{EMap}_i : \mathsf{Dias}_i(\phi) \cup \mathsf{Boxs}_i(\phi) \mapsto \mathsf{ESet}_i \setminus \mathsf{Vars}_i(\phi)$.

4.4. Matrix

We define the matrix of the translation as a function, $\mathsf{TransMtx}(\phi)$, which uses the sets defined in the prefix to translate a modal formula ϕ into four sets of propositional formulae. The first set is a top level propositional mapping of the formula at depth 0. The second and third sets define the logic for the diamonds and boxes respectively. The fourth set defines the conditions for the witness variables to be required true: that if any diamond at depth i is to be branched in to, then there is a witness at depth $i+1$ and therefore the contents of all selected boxes at depth i must always be true.

First we define the function $\mathsf{PMap}_i(\varphi)$ which takes a modal formula φ at depth i and returns a propositional representation of it. That is, it returns the formula with each of its topmost boxes and diamonds replaced with their existential variable mappings by way of $\mathsf{EMap}_i(\varphi)$ and is defined as follows:

- if φ is a propositional variable, then $\mathsf{PMap}_i(\varphi) = \varphi$
- if φ is $\Diamond\varphi_1$ or $\Box\varphi_1$, then $\mathsf{PMap}_i(\varphi) = \mathsf{EMap}_i(\varphi)$
- if φ is $\neg\varphi_1$ then $\mathsf{PMap}_i(\varphi) = \neg\mathsf{PMap}_i(\varphi_1)$
- if φ is $\varphi_1 \wedge \varphi_2$ then $\mathsf{PMap}_i(\varphi) = \mathsf{PMap}_i(\varphi_1) \wedge \mathsf{PMap}_i(\varphi_2)$
- if φ is $\varphi_1 \vee \varphi_2$ then $\mathsf{PMap}_i(\varphi) = \mathsf{PMap}_i(\varphi_1) \vee \mathsf{PMap}_i(\varphi_2)$

Next, we introduce three sets of definitions that link the variables in $\mathsf{PMap}_0(\phi)$ to the sub-formulae they have replaced via suitable implications, DiaMap^ϕ for diamonds and BoxMap^ϕ for boxes.

$$\mathsf{DiaMap}^\phi = \bigwedge_{i=0}^{\mathsf{Depth}(\phi)-1} \bigwedge_{\Diamond\varphi \in \mathsf{Dias}_i(\phi)} \left((\mathsf{EMap}_i(\Diamond\varphi) \wedge \bigwedge_{x \in \mathsf{UMap}_i(\Diamond\varphi)} x \wedge \bigwedge_{y \in \mathsf{USet}_i \setminus \mathsf{UMap}_i(\Diamond\varphi)} \neg y) \rightarrow \mathsf{PMap}_{i+1}(\varphi) \right)$$

$$\mathsf{BoxMap}^{\phi} = \bigwedge_{i=0}^{\mathsf{Depth}(\phi)-1} \bigwedge_{\Box\varphi \in \mathsf{Boxs}_i(\varphi)} \big((wit_{i+1} \wedge \mathsf{EMap}_i(\Box\varphi)) \rightarrow \mathsf{PMap}_{i+1}(\varphi)\big)$$

$$\mathsf{WitMap}^{\phi} = \bigwedge_{i=0}^{\mathsf{Depth}(\phi)-1} \bigwedge_{\Diamond\varphi \in \mathsf{Dias}_i(\phi)} (\mathsf{EMap}_i(\Diamond\varphi) \rightarrow wit_{i+1})$$

DiaMap^{ϕ} says that if a diamond has been selected and its enumeration over USet_{i+1} is true, then its contents are true. BoxMap^{ϕ} says that if a box is selected and any diamond at its depth has been selected, making its $witness$ true via WitMap^{ϕ}, then its contents are true. WitMap^{ϕ} says that if any diamond at depth i is selected, there is a witness at depth $i+1$ for all selected boxes.

Then,

$$\mathsf{TransMtx}(\phi) = \mathsf{PMap}_0(\phi) \wedge \mathsf{DiaMap}^{\phi} \wedge \mathsf{BoxMap}^{\phi} \wedge \mathsf{WitMap}^{\phi}$$

and the entire translated QBF $\mathsf{Trans}(\phi)$ of a formula φ in $\mathbf{K}_{NNF}$ satisfying the above mentioned uniqueness constraint is

$$\mathsf{Trans}(\phi) = \exists\mathsf{ESet}_0 \forall\mathsf{USet}_1 \exists\mathsf{ESet}_1 ... \forall\mathsf{USet}_d \exists\mathsf{ESet}_d.\mathsf{TransMtx}(\phi).$$

4.5. Optimised Prefix

We can optimise the prefix order of the translation with regards to the polynomial hierarchy by shifting some existential variables to the innermost existential quantifier. Doing so results in a translation which may have a shorter prefix: if all existential variables in a quantifier are shifted to the innermost existential quantifier, then the two surrounding universal quantifiers can be merged.

All propositional variables in ϕ can be shifted to the innermost existential prefix in ϕ' without affecting satisfiability: While the truth value of propositional variables is dependent on universal variables in ϕ', none of the universal variables are dependent on the truth of propositional variables. To put this in terms of a tableaux search: the only assignments at a node in a tableaux search which govern the formulae in a child of that node are assignments to boxes and diamonds. By modifying the prefix in this way, ϕ' now simulates a tableaux search in which all assignments to propositional variables in a branch are delayed until the leaf of that branch. Furthermore, we can shift the existential variables representing diamonds and boxes which *occur deterministically* to the innermost existential quantifier of ϕ'.[1] A diamond or box at depth i occurs deterministically if it does not occur within the scope of a disjunction *at that depth*. For example $\Diamond a$ occurs deterministically at depth 1 in $\Box b \vee \Diamond(\Diamond a \wedge b)$ but does not occur deterministically in $\Diamond((\Diamond a \wedge b) \vee \Box b)$. The truth of diamonds and boxes which occur deterministically at depth i is a direct consequence of the truth of their parents at depth $i-1$, therefore their corresponding universal variables are only dependent on the truth of their parent.

Let $\mathsf{DDias}_i(\phi)$ and $\mathsf{DBoxs}_i(\phi)$ be the set of boxes and diamonds that occur deterministically at depth i in ϕ. Then our optimised prefix is defined as follows:

$$\exists\mathsf{E}^{+}\mathsf{Set}_0 \forall\mathsf{USet}_1 \exists\mathsf{E}^{+}\mathsf{Set}_1 ... \forall\mathsf{USet}_d \exists\mathsf{E}^{+}\mathsf{Set}_d,$$

where, for $0 \leq i < d$, the optimised existential sets are defined as follows:

$$\begin{aligned}
\mathsf{E}^{+}\mathsf{Set}_i &= \{\mathsf{EMap}_i(\varphi) \mid \varphi \in \{\mathsf{Dias}_i(\phi) \cup \mathsf{Boxs}_i(\phi)\} \setminus \{\mathsf{DDias}_i(\phi) \cup \mathsf{DBoxs}_i(\phi)\}\} \\
\text{and} \\
\mathsf{E}^{+}\mathsf{Set}_d &= \mathsf{Vars}_0(\phi) \cup \mathsf{Vars}_1(\phi) \cup ... \cup \mathsf{Vars}_d(\phi) \cup \mathsf{Wit}^{\phi} \cup \\
&\quad \mathsf{DMaps}_0(\phi) \cup \mathsf{DMaps}_1(\phi) \cup ... \cup \mathsf{DMaps}_{d-1}(\phi) \\
\text{where} \\
\mathsf{DMaps}_i(\phi) &= \{\mathsf{EMap}_i(\varphi) | \varphi \in \{\mathsf{DDias}_i(\phi) \cup \mathsf{DBoxs}_i(\phi)\}\}.
\end{aligned}$$

We use $\mathsf{Trans}^{+}(\phi)$ to refer to the optimised translation of a formula ϕ, i.e., the one using the optimised prefix.

[1] We can in fact completely remove these existential variables entirely from the prefix by modifying the matrix definition slightly. For the sake of simplicity we keep the existential variables so that the matrix definition remains the same

4.6. Correctness

We very briefly sketch proofs here of the correctness of our translations. For full proofs, see the extended version of this paper available at https://gitlab.cs.man.ac.uk/j13280mh1/qbf-reasoning-for-modal-logic.

Proposition 1. *The QBF translation Trans(ϕ) of a modal formula ϕ is true iff ϕ is satisfiable.*

For the forward direction, we create a tableaux data structure of ϕ (representing a tree model) using the semantics of K and create a mapping to a corresponding winning strategy data structure (representing a QBF model) of $\mathsf{Trans}(\phi)$ which satisfies $\mathsf{Trans}(\phi)$ by QBF semantics. The reverse direction maps a winning strategy of $\mathsf{Trans}(\phi)$ to a tableaux of ϕ in the same way.

Proposition 2. *The QBF translation Trans$^+(\phi)$ of a modal formula ϕ is true iff Trans(ϕ) is true.*

We map a winning strategy of $\mathsf{Trans}^+(\phi)$ to a winning strategy of $\mathsf{Trans}(\phi)$ and vice versa, using the matrix definition - which is the same in both translations - to show that shifting the prefix in $\mathsf{Trans}^+(\phi)$ does not meaningfully remove constraints on winning strategies.

Proposition 3. *Trans$^+(\phi)$ can be computed in time polynomial in the length ϕ.*

For each requirement of $\mathsf{Trans}^+(\phi)$, a polynomial-time function is outlined.

Corollary 1. *Trans$^+(\cdot)$ is a polynomial-time, satisfiability preserving translation from $\mathbf{K}$ to QBF.*

4.7. Optimised QBF to K

It should be clear that, for any modal formula ϕ, the prefix of $\mathsf{Trans}^+(\phi)$ has at most as many alternations as the prefix of $\mathsf{Trans}(\phi)$ and may have fewer; it is thus closer to a polynomial-hierarchy-preserving translation. It also illustrates that disjunctions between only propositional variables do not affect the alternation depth of a modal formula. We can demonstrate our claim of optimality with regards to the polynomial hierarchy by defining an optimised translation from closed QBF to K such that our optimised K to QBF translation respects the alternation depth of Q_{PNF} formulae translated with our optimised QBF to K translation: This shows that, for these formulae, $\mathsf{Trans}^+(\phi)$ polynomially extracts the correct class of the polynomial hierarchy while $\mathsf{Trans}(\phi)$ does not.[2]

We define $\mathsf{QTrans}^+(\cdot)$ as an optimisation of the Schmidt-Smolka translation of QBF to $\mathcal{ALC}$ concepts [4]. $\mathsf{QTrans}^+(\cdot)$ translates *simplified* closed CNF Q_{PNF} formulae to $\mathbf{K}$ formulae in NNF, preserving QBF truth to $\mathbf{K}$ satisfiability.[3] $\mathsf{QTrans}^+(\cdot)$ is defined such that, for all such Q_{PNF} formulae ψ, $\mathsf{AltD}(\mathsf{Trans}^+(\mathsf{QTrans}^+(\psi))) \leq \mathsf{AltD}(\psi)$ where $\mathsf{AltD}(\psi)$ is the alternation depth of a Q_{PNF} formula ψ.[4]

For a simplified closed CNF Q_{PNF} formula $\psi = P.(C_1 \wedge C_2 \wedge \cdots \wedge C_n)$ where P is a prefix and C_i are *clauses* (disjunctions of literals), we define a function Alt_ψ that returns the *alternation number* of a literal in the prefix of ψ. We number the alternations of quantifiers in a prefix from $\forall$ to $\exists$ from 1 up from left to right; if the outermost quantifier is existential, its variables are numbered 0. For example, if $P = \exists\{e1, e2\}\forall\{u1, u2\}\exists\{e3\}\forall\{u3\}\exists\{e4\}$ then $\mathsf{Alt}_\psi(e1) = \mathsf{Alt}_\psi(e2) = 0$, $\mathsf{Alt}_\psi(u1) = \mathsf{Alt}_\psi(u2) = \mathsf{Alt}_\psi(e3) = 1$, $\mathsf{Alt}_\psi(u3) = \mathsf{Alt}_\psi(e4) = 2$. We also define the Universal Count function UC_ψ to return the number of universally quantified variables in an alternation. For example, for the P above, $\mathsf{UC}_\psi(0) = 0$, $\mathsf{UC}_\psi(1) = 2$, $\mathsf{UC}_\psi(2) = 1$. $\mathsf{QTrans}^+(\cdot)$ assumes that all literals in clauses are ordered from left to right in ascending order of Alt_ψ.

[2] We note that the translation presented in [18] does not translate modal formulae to PNF, but when converted to PNF the formulae have the same alternation depth as the unoptimised $\mathsf{Trans}(\phi)$

[3] By simplified, we mean the well known, polynomial-time simplification of CNF closed QBF formulae with an innermost universal quantifier: in this case the innermost quantifier can be removed by removing from each clause any occurrences of universal variables quantified on the innermost quantifier. The translation assumes, without loss of generality, that the innermost quantifier of input formulae is always existential.

[4] See the extended version of this paper available at: https://gitlab.cs.man.ac.uk/j13280mh1/qbf-reasoning-for-modal-logic for a proof. The alternation depth can decrease when variables in different quantification levels in the prefix of ψ never interact in its matrix, thus the prefix can be simplified. E.g., the prefix of $\psi = \forall\{a_1\}\exists\{e1\}\forall\{a2\}\exists\{e2\}.((a_1 \vee e_1) \wedge (a_2 \vee e_2))$ can be simplified to $\forall\{a_1, a_2\}\exists\{e_1, e_2\}$ and $\mathsf{Trans}^+(\mathsf{QTrans}^+(\psi))$ will result in a prefix of $\forall\exists$ rather than $\forall\exists\forall\exists$.

$\mathsf{QTrans}^+(\psi) = \mathsf{QPT}(P) \wedge \mathsf{QCT}(C_1, 0) \wedge \mathsf{QCT}(C_2, 0) \wedge \cdots \wedge \mathsf{QCT}(C_n, 0)$ for functions *QBF Prefix Translate* QPT and *QBF Clause Translate* QCT, defined below.

For non-empty sets of variables, E and U, and a prefix beginning with $\exists$, P':

- $\mathsf{QPT}(\exists E \forall U.P') = \mathsf{QPT}(\forall U.P')$
- $\mathsf{QPT}(\exists E) = \varepsilon$
- if $u \in U$, then $\mathsf{QPT}(\forall U.P') = \Diamond\Box^{|U\setminus\{u\}|}u \wedge \Diamond\Box^{|U\setminus\{u\}|}\neg u \wedge \Box(\mathsf{QPT}(\forall U \setminus \{u\}.P'))$
- if P' contains $\forall$, then $\mathsf{QPT}(\forall\{u\}.P') = \Diamond u \wedge \Diamond\neg u \wedge \Box(\mathsf{QPT}(P'))$; otherwise $\mathsf{QPT}(\forall\{u\}.P') = \Diamond u \wedge \Diamond\neg u$

For a literal l, a non-empty clause C', and integer i:

- $\mathsf{QCT}(l \vee C', i) = l \vee \mathsf{QCT}(C', i)$ if $i = \mathsf{Alt}_\psi(l)$
- $\mathsf{QCT}(l, i) = l$ if $i = \mathsf{Alt}_\psi(l)$
- $\mathsf{QCT}(l \vee C', i) = \Box^{UC_\psi(i+1)}(\mathsf{QCT}(l \vee C', i+1))$ if $i < Alt_\psi(l)$
- $\mathsf{QCT}(l, i) = \Box^{\mathsf{UC}_\psi(i+1)}(\mathsf{QCT}(l, i+1))$ if $i < \mathsf{Alt}_\psi(l)$

5. Implementation and Evaluation

We implemented the translation $\mathsf{Trans}^+(\phi)$ as described above, converting $\mathbf{K}$ formulae in InToHyLo file format to closed QBF in QDIMACS format, two common formats used by modal logic and QBF reasoners respectively. Since the translation requires $\mathbf{K}$ formulae in NNF, our implementation first applies standard De Morgan identities on input formulae to convert them to NNF but does not currently support bi-implication. The QDIMACS format describes formulae in CNF. We convert the $\mathsf{Trans}^+(\phi)$ to an equisatisfiable CNF formula in the usual way, i.e., by recursively introducing existentially quantified variable names for $\wedge$ sub-formulae within $\vee$ sub-formulae, replacing these $\wedge$ formula with their name in the $\vee$ sub-formula and then introducing clauses for implications from the name to each constituent sub-formula of the $\wedge$ sub-formula. We found that the optimised translation performed slightly better on average than the un-optimised translation, though both performed similarly.

Since $\mathsf{Trans}^+(\phi)$ requires an extraction of sets of diamonds and boxes at each depth in a formula, our implementation requires a comparison operation between diamonds and boxes. It uses a lazy comparison which does not consider the commutativity of $\wedge$ and $\vee$ and so, for example, $\Box(a \vee b \vee c)$ and $\Box(c \vee b \vee a)$ are considered to be two different boxes. While an implementation considering commutativity would still be polynomial, it would require an ordering of formulae which would considerably slow down the translation for possibly little gain.

We use the QBF reasoner DepQBF to solve translated formulae. While based on QCDCL, DepQBF contains many more QBF-specific optimisations such as a counter-part to CDCL, *solution driven cube learning* or SDCL and various variable dependency optimisations giving the reasoner its namesake [**?**]. We disable most of these optimisations, not only to isolate the effects of QCDCL but also because we found most of them to slow down performance on the majority of translated modal formulae. The specific parameters used were: –no-sdcl –no-qbce-dynamic –no-dynamic-nenofex –no-empty-formula-watching –no-trivial-falsity –no-trivial-truth. Out of these, disabling SDCL improved performance most, though DepQBF still performs well without changing any parameters. Two optimisations that were crucial to performance were QCDCL and pure literal watching. Disabling either of these, through the parameters –no-cdcl and –no-pure-literals respectively, would cause the reasoner to timeout on many formulae which could otherwise be solved in less than a second without disabling them. Disabling CDCL causes DepQBF to fall back on a QDPLL algorithm, backtracking chronologically from conflicts; that we can isolate the effects of QCDCL within DepQBF is evidence that the results shown in the following sections are indeed caused by the QCDCL algorithm rather than some other optimisations specific to DepQBF.

We compare DepQBF with two modal logic reasoners: CegarBOX [12] and Spartacus [24]. CegarBOX is the more recent reasoner, based on counter example guided abstraction refinement and using an

oracle SAT solver to solve propositional abstractions of modal formulae. CegarBOX claims to be the fastest modal logic reasoner and, to our knowledge, this claim has not been challenged to date. Spartacus is an older, more traditional tableaux reasoner, though still performs competitively against other modal logic reasoners [12]. Both CegarBOX and Spartacus were run in their default configurations, using the latest versions of both reasoners publicly available to date.

5.1. Test Details

We compared the reasoners on four test sets, two of which were taken from CegarBOX's evaluation, and originating from Nalon et. al. [25], though with some important caveats. These sets are the randomly generated 3CNF K benchmarks and the extended LWB K-Benchmarks. The 3CNF K benchmarks are a set of 1,000 randomly generated formulae generated through various configurations according to a standard method from Patel et. al [16]. However, the configurations used for these benchmarks do not go far in exploiting the method of Patel et. al., with formulae being of total depth 1 or 2. We will henceforth call these the KSP_3CNF benchmarks.

We were only able to test a small subset of the extended LWB K-Benchmarks. The extended LWB K-Benchmarks are procedurally generated with classes of satisfiable and unsatisfiable formulae based on the procedures described by Balsiger et. al. [26]. Formulae in this set are marked with "p" for provable - valid formulae which are negated as to be unsatisfiable - and "n" for not provable - invalid formulae negated to be satisfiable. The formulae in this set can be extremely large, in some cases exceeding 80MB for a single formula. Unfortunately, our implementation of the translation struggles on large formulae with many modal atoms, making translating the entire set unreasonable. We therefore use a small subset of this test set, only including formulae up to 1.2MB and without bi-implications. We used the LWB generator of CegarBOX [5] however, some formulae also failed to generate. We were left with a total of 382 formulae from that set, which we will henceforth call LWB_small. Our implementation is far from optimised, and perhaps a future implementation could allow a larger set of these formulae to be translated; however, this small subset still contains very hard formulae and formulae with high modal depth. Furthermore, we can observe trends in the performance of DepQBF on classes of formulae from this set which may give us an idea of how it could perform on the larger formulae of each class.

In addition to the extended LWB and random 3CNF benchmarks, CegarBOX's evaluation also uses MQBF benchmarks, which we have not tested in our evaluation [17]. These are randomly generated QBF formulae which are translated to modal logic using the Schmidt-Smolka translation [4]. We are interested in comparing purely modal benchmarks using a QBF reasoner and it is already known that QBF reasoners perform better on the original QBF formulae than modal reasoners do on the translated formualae [17].

The other two sets of formulae we used are also randomly generated CNF formulae using the method of Patel et. al. [16]. The first configuration, henceforth referred to as Config 1, uses the parameters d=3, m=1, N = 5, C = [[0,2,1]], p = [[],[0,3,0],[0,3,3,0]] from Figure 12 of their paper. The second configuration, henceforth referred to as Config 2, uses the parameters d = 5, m = 1, N = 5, c = [[1,8,1],[1,2]], p = [[[1,0],[0,1,0],[0,1,1,0]],[[1,0],[0,1,0]]], from Figure 15 of their paper. d is the depth, m the number of modalities, N the number of propositional variables, C the probability distribution for clause sizes and p is the probability distribution for the number of propositional variables in a clause. For each of these configurations, we generated 15 formulae for clause-variable ratios (cvr) from 5 to 150 in intervals of 5 for a total of 450 formulae per set. The idea behind the randomly generated tests is that, by varying the clause variable ratio, the difficulty of the generated formulae vary. Higher clause variable ratios have a higher probability of being unsatisfiable; in theory, the hardest formulae generated are those generated at the clause variable ratio in which there is an even chance of generated formulae being satisfiable or unsatisfiable, the so-called *phase transition* [27]. Therefore, one would expect a pattern of fast-slow-fast from each of the reasoners on these formulae.

For the KSP_3CNF test set and the LWB_small test set, we run each of our reasoners with a timeout

[5] available at https://github.com/cormackikkert/LWB-benchmark-generator

of 10 seconds and record the cumulative sum of instances solved in intervals of 0.1, 0.2, 0.5, 1, 2, 4, 8, 10 in accordance with the methodology of CegarBOX's evaluation [12]. For Config 1 and 2, we plot the median solve time at each clause variable ratio for each reasoner with a timeout of 30 seconds, in accordance with the methodology of Patel et. al. [16]. All tests were run on a computer running Ubuntu 22.04.5 LTS with an Intel Core i5-1335U, 16GB RAM with each reasoner being the only major process running whenever they are running. We do not include the translation times in the timing of DepQBF as we are aiming to compare the QBF reasoner itself with the modal reasoners. [6]

5.2. Results

Figure 1 shows the results of our testing on the LWB_small test set and the KSP_3CNF test set. On the LWB_small test set, it appears that DepQBF and CegarBOX are competitive - both performing better than the tableaux-based Spartacus - with CegarBOX performing slightly better than DepQBF. Out of 382 formulae, CegarBOX solved 294 while DepQBF solved 283. First, that both reasoners timed out on so many formulae is a demonstration that many formulae in the LWB test set do not require a large size to be hard for reasoners and that it is likely that many large formulae in the extended LWB set are quite easy. However, presenting the results in this way obscures some interesting details: the classes of formulae that each reasoner found hard are in some cases very different: for example, DepQBF found all 53 k_branch_p formulae in the set unsatisfiable in less than 0.2 seconds whereas CegarBOX timed out on 26 of them. On the other hand, DepQBF could only find 4 of 14 k_d4_n formulae satisfiable in the time limit while CegarBOX solved them all within 0.3 seconds. Spartacus also solved all k_d4_n formulae, though much slower than CegarBOX with the largest formula taking over 9 seconds to solve. Some classes of formula were easy for both reasoners. For example, both solved all 22 k_path_n formulae in the set in less than 0.3 seconds. In the extended LWB set, there are 56 of these, going up to 80MB; in short, combining these classes of formulae in one set does not give a good showcase of the difference in performance of reasoners. It is unclear how DepQBF would perform on the larger set, but trends in some classes of formulae suggest that some of the larger formulae may not be a challenge.

The situation is similar for the KSP_3CNF test on the right of Figure 1: On the surface, DepQBF and CegarBOX are close in overall performance and far ahead of Spartacus, with DepQBF this time being slightly better: Out of 1000 formulae, CegarBOX solved 888 of them within the time limit compared to DepQBF solving 906 of them. But this presentation again obscures the true differences in the reasoners' performance: In this case, there is one particular configuration of formulae that DepQBF found very hard but were not a challenge for CegarBOX and the majority of the other formulae were easy for DepQBF. First, note that within 0.1 seconds DepQBF already solves the majority of the test set. Of the 94 formulae DepQBF timed out on, 69 were from the configuration d=1, m=1, N=6 c = [[0,0,1]] p = [[[],[],[1,0,0,0]]], CegarBOX did not time out on any of these. These formulae have a modal depth of just one with clauses of uniform size three with no propositional variables generated at depth 0. Their challenge lies in solving many configurations of purely propositional 3CNF formulae at depth 1. Given this, it may be unsurprising that CegarBOX, using an oracle SAT solver, performs very well on this configuration. This configuration of random formulae was criticised by Horrocks, Patel et. al. for its focus on propositional difficulty, while ignoring modal difficulty, leading them to create the extensible generation method responsible for the random test sets [15, 16]. In Figure 2, we test two configurations of formulae directly from [16] with depth 3 and 5 for Config 1 and 2, respectively.

Looking at Figure 2, it is immediately clear that DepQBF trivialises both configurations of formulae for all clause-variable ratios, vastly outperforming both modal reasoners. First, while these graphs only show median run times, DepQBF did not time out on any of these formulae. In fact, every formula was solved in under roughly 3 seconds aside from two outliers in Config 2 taking more than 10 seconds. CegarBOX had 50 timeouts in Config 1 and 272 in Config 2 while Spartacus had 168 in Config 1 and 76 in Config 2. These timeouts explain the spikes in median run time for CegarBOX and Spartacus in Config 1 and 2, respectively. The behaviour of Spartacus on Config 2 is, however, surprising: while

[6] All tested formulae can be found in https://gitlab.cs.man.ac.uk/j13280mh1/qbf-reasoning-for-modal-logic

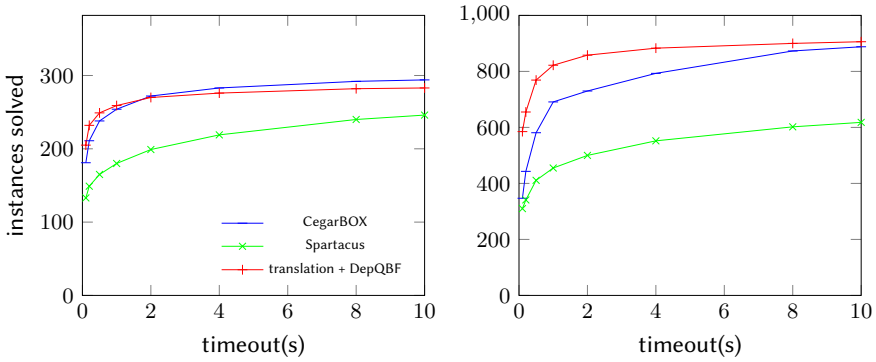

Figure 1: Cumulative instances solved in a timeout of 10 seconds for CegarBOX, Spartacus and DepQBF on the test sets LWB_small (left) and KSP_3CNF (right).

CegarBOX times out on almost every formula from a cvr of 60, Spartacus solves some formulae very fast while timing out on others. From a cvr of 85, the majority of generated formulae in Config 2 are unsatisfiable. Spartacus either finds them unsatisfiable very quickly or times out, while also timing out on all formulae which are satisfiable from that cvr on. CegarBOX, on the other hand, seems to struggle with randomly generated formulae of high depth.

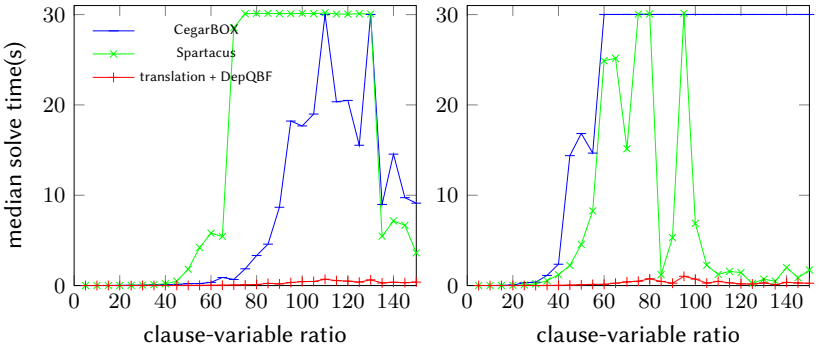

Figure 2: Median run time of CegarBOX, Spartacus and DepQBF for clause-variable-ratios from 5 to 150 over randomly generated formulae of Config 1 (left) and 2 (right).

6. Conclusion / Future Work

We have presented an optimised translation from $\mathbf{K}$ to QBF, $\mathsf{Trans}^{+}(\cdot)$, with consideration to the polynomial hierarchy, demonstrated by defining classes of formulae, $\mathsf{QTrans}^{+}(\psi)$, for which the translation correctly extracts their alternation depth. Using this translation, we have shown that a QCDCL-based QBF reasoner can perform very well and in some cases vastly outperform native modal logic reasoners on modal logic formulae, rendering some sets of modal formulae which are difficult for native reasoners trivial, showing the potential of QBF reasoning techniques in modal/description Logic. Extending the translation to multiple modalities/roles is straightforward, only requiring a separate set of witness variables for each modality, and a slight adjustment to the matrix to ensure witness variables are only implied when a diamond of their modality has been branched into, rather than just selected.

Given our empirical results, an obvious further direction is to devise a native counterpart to QCDCL for Modal/Description logic. Another point of interest is to translate other PSPACE-complete Modal/Description logics to QBF, such as $\mathcal{ALCI}$. There is also some more to be done in further optimising $\mathsf{Trans}^{+}(\cdot)$: as it is presented, it considers formulae only in NNF, without requiring any alternation-specific normalisation of formulae; $\mathsf{QTrans}^{+}(\cdot)$ outlines a potential *alternation normal form* of $\mathbf{K}$ formulae for which a more tailored, optimal translation could be defined.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] R. E. Ladner, The computational complexity of provability in systems of modal propositional logic, SIAM-JOC 6 (1977) 467–480.

[2] J. Y. Halpern, Y. Moses, A guide to completeness and complexity for modal logic of knowledge and belief, AI 54 (1992) 319–379.

[3] H. Kleine Büning, M. Karpinski, A. Flogel, Resolution for quantified boolean formulas, Information and computation 117 (1995) 12–18.

[4] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, Artificial intelligence 48 (1991) 1–26.

[5] K. Schild, A correspondence theory for terminological logics: Preliminary report, Techn. Univ., 1991.

[6] K. Schild, Terminological cycles and the propositional μ-calculus, in: Principles of Knowledge Representation and Reasoning, Elsevier, 1994, pp. 509–520.

[7] G. De Giacomo, M. Lenzerini, Boosting the correspondence between description logics and propositional dynamic logics, in: AAAI, volume 94, 1994, pp. 205–212.

[8] I. Horrocks, U. Hustadt, U. Sattler, R. Schmidt, Computational modal logic, in: P. Blackburn, J. Van Benthem, F. Wolter (Eds.), Handbook of Modal Logic, volume 3 of *Studies in Logic and Practical Reasoning*, Elsevier, 2007, pp. 181–245. URL: https://www.sciencedirect.com/science/article/pii/S1570246407800073. doi:https://doi.org/10.1016/S1570-2464(07)80007-3.

[9] J. M. Silva, K. A. Sakallah, Grasp-a new search algorithm for satisfiability, in: Proceedings of International Conference on Computer Aided Design, IEEE, 1996, pp. 220–227.

[10] L. Zhang, S. Malik, Conflict driven learning in a quantified boolean satisfiability solver, in: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, 2002, pp. 442–449.

[11] U. Hustadt, R. Schmidt, An empirical analysis of modal theorem provers, Journal of Applied Non-Classical Logics 9 (1999) 479–522. doi:10.1080/11663081.1999.10510981.

[12] R. Goré, C. Kikkert, Cegar-tableaux: improved modal satisfiability via modal clause-learning and sat, in: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Springer, 2021, pp. 74–91.

[13] J.-M. Lagniez, D. Le Berre, T. De Lima, V. Montmirail, A recursive shortcut for cegar: Application to the modal logic k satisfiability problem., in: IJCAI, 2017, pp. 674–680.

[14] U. Hustadt, R. A. Schmidt, An empirical analysis of modal theorem provers, Journal of Applied Non-Classical Logics 9 (1999) 479–522.

[15] I. Horrocks, P. F. Patel-Schneider, R. Sebastiani, An analysis of empirical testing for modal decision procedures, Logic Journal of IGPL 8 (2000) 293–323.

[16] P. F. Patel-Schneider, R. Sebastiani, A new general method to generate random modal formulae for testing decision procedures, Journal of Artificial Intelligence Research 18 (2003) 351–389.

[17] F. Massacci, F. M. Donini, Design and results of tancs-2000 non-classical (modal) systems comparison, in: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Springer, 2000, pp. 52–56.

[18] G. Pan, M. Y. Vardi, Optimizing a BDD-based modal solver, in: CADE-03, volume 2741 of *LNAI*, SV, 2003.

[19] R. Sebastiani, M. Vescovi, Automated reasoning in modal and description logics via sat encoding: the case study of k (m)/alc-satisfiability, Journal of Artificial Intelligence Research 35 (2009) 343–389.

[20] M. Kaminski, T. Tebbi, Inkresat: modal reasoning via incremental reduction to sat, in: International Conference on Automated Deduction, Springer, 2013, pp. 436–442.
[21] F. Lonsing, U. Egly, Depqbf 6.0: A search-based qbf solver beyond traditional qcdcl, in: International Conference on Automated Deduction, Springer, 2017, pp. 371–384.
[22] A. Das, S. Marin, Modal logic and the polynomial hierarchy: from qbfs to k and back., in: AiML, 2022, pp. 329–348.
[23] C. Areces, R. Gennari, J. Heguiabehere, M. de Rijke, Tree-based heuristics in modal theorem proving, in: Proceedings of ECAI'2000, 2000, pp. 199–203.
[24] D. Götzmann, M. Kaminski, G. Smolka, Spartacus: A tableau prover for hybrid logic, Electronic Notes in Theoretical Computer Science 262 (2010) 127–139.
[25] C. Nalon, U. Hustadt, C. Dixon, : A resolution-based prover for multimodal k, in: International Joint Conference on Automated Reasoning, Springer, 2016, pp. 406–415.
[26] P. Balsiger, A. Heuerding, S. Schwendimann, A benchmark method for the propositional modal logics k, kt, s4, Journal of Automated Reasoning 24 (2000) 297–317.
[27] P. Cheeseman, Where the really hard problems are, in: International Joint Conference on Artificial Intelligence, 1991.