Encoding the satisfiability of modal and description logics into SAT: the case study of K(m)/$\mathcal{ALC}$

Roberto Sebastiani and Michele Vescovi

DIT, Università di Trento, Via Sommarive 14, I-38050, Povo, Trento, Italy.
rseba@dit.unitn.it, vescovi@dit.unitn.it

Abstract. In the last two decades, modal and description logics have been applied to numerous areas of computer science, including artificial intelligence, formal verification, database theory, and distributed computing. For this reason, the problem of automated reasoning in modal and description logics has been throughly investigated.
In particular, many approaches have been proposed for efficiently handling the satisfiability of the core normal modal logic K_m, and of its notational variant, the description logic $\mathcal{ALC}$. Although simple in structure, K_m/$\mathcal{ALC}$ is computationally very hard to reason on, its satisfiability being PSPACE-complete.
In this paper we explore the idea of encoding K_m/$\mathcal{ALC}$-satisfiability into SAT, so that to be handled by state-of-the-art SAT tools. We propose an efficient encoding, and we test it on an extensive set of benchmarks, comparing the approach with the main state-of-the-art tools available.
Although the encoding is necessarily worst-case exponential, from our experiments we notice that, in practice, this approach can handle most or all the problems which are at the reach of the other approaches, with performances which are comparable with, or even better than, those of the current state-of-the-art tools.

1 Introduction

In the last two decades, modal and description logics have been applied to numerous areas of computer science, including artificial intelligence, formal verification, database theory, and distributed computing. For this reason, the problem of automated reasoning in modal and description logics has been throughly investigated (see, e.g., [2, 9, 6, 10]). Many approaches have been proposed for efficiently handling the satisfiability of modal and description logics, in particular of the core normal modal logic K_m and of its notational variant, the description logic $\mathcal{ALC}$ (see, e.g., [2, 10, 3, 4, 7, 8, 1, 12]). Notice that, although simple in structure, K_m/$\mathcal{ALC}$ is computationally very hard to reason on, as its satisfiability is PSPACE-complete [6].

In this paper we explore the idea of encoding K_m/$\mathcal{ALC}$-satisfiability into SAT, so that to be handled by state-of-the-art SAT tools. We propose an efficient encoding, with four simple variations. We test (the four variations of) it on an extensive set of benchmarks, comparing the results with those of the main state-of-the-art tools for K_m-satisfiability available. Although the encoding is necessarily worst-case exponential (unless PSPACE = NP), from our experiments we notice that, in practice, this approach can handle most or all the problems which are at the reach of the other approaches, with performances which are comparable with, or even better than, those of the current state-of-the-art tools.

For lack of space, in this short version of the paper we omit the proof of Theorem 1, the description of the empirical tests and results, and every reference to related work. All such information can be found in the extended version of the paper [13], which is publicly downloadable [1].

2 Background

We recall some basic definitions and properties of K_m. Given a non-empty set of primitive propositions $\mathcal{A} = \{A_1, A_2, \ldots\}$ and a set of m modal operators $\mathcal{B} = \{\Box_1, \ldots, \Box_m\}$, the language of K_m is the least set of formulas containing $\mathcal{A}$, closed under the set of propositional connectives $\{\neg, \wedge\}$ and the set of modal operators in $\mathcal{B}$. Notationally, we use the Greek letters $\alpha, \beta, \varphi, \psi, \nu, \pi$ to denote formulas in the language of K_m (K_m-formulas hereafter). We use the standard abbreviations, that is: "$\Diamond_r \varphi$" for "$\neg \Box_r \neg \varphi$", "$\varphi_1 \vee \varphi_2$" for "$\neg(\neg\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \rightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \leftrightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2) \wedge \neg(\varphi_2 \wedge \neg\varphi_1)$", "$\top$" and "$\bot$" for the constants "true" and "false". (Hereafter formulas like $\neg\neg\psi$ are implicitly assumed to be simplified into ψ, so that, if ψ is $\neg\phi$, then by "$\neg\psi$" we mean "ϕ".) We call *depth* of φ, written *depth*(φ), the maximum number of nested modal operators in φ. We call a *propositional atom* every primitive proposition in $\mathcal{A}$, and a *propositional literal* every propositional atom (*positive literal*) or its negation (*negative literal*).

In order to make our presentation more uniform, we adopt from [2, 10] the representation of K_m-formulas from the following table:

α	α_1	α_2	β	β_1	β_2	π^r	π^r_0	ν^r	ν^r_0
$(\varphi_1 \wedge \varphi_2)$	φ_1	φ_2	$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2	$\Diamond_r \varphi_1$	φ_1	$\Box_r \varphi_1$	φ_1
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$	$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$	$\neg\Box_r \varphi_1$	$\neg\varphi_1$	$\neg\Diamond_r \varphi_1$	$\neg\varphi_1$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$	$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2				

in which non-literal K_m-formulas are grouped into four categories: α's (conjunctive), β's (disjunctive), π's (existential), ν's (universal).

A *Kripke structure* for K_m is a tuple $\mathcal{M} = \langle \mathcal{U}, \mathcal{L}, \mathcal{R}_1, \ldots, \mathcal{R}_m \rangle$, where $\mathcal{U}$ is a set of states, $\mathcal{L}$ is a function $\mathcal{L} : \mathcal{A} \times \mathcal{U} \longmapsto \{True, False\}$, and each $\mathcal{R}_r$ is a binary relation on the states of $\mathcal{U}$. With an abuse of notation we write "$u \in \mathcal{M}$" instead of "$u \in \mathcal{U}$". We call a *situation* any pair $\mathcal{M}, u$, $\mathcal{M}$ being a Kripke structure and $u \in \mathcal{M}$. The binary relation $\models$ between a modal formula φ and a situation $\mathcal{M}, u$ is defined as follows:

$$
\begin{array}{lll}
\mathcal{M}, u \models A_i,\ A_i \in \mathcal{A} & \Longleftrightarrow & \mathcal{L}(A_i, u) = True; \\
\mathcal{M}, u \models \neg A_i,\ A_i \in \mathcal{A} & \Longleftrightarrow & \mathcal{L}(A_i, u) = False; \\
\mathcal{M}, u \models \alpha & \Longleftrightarrow & \mathcal{M}, u \models \alpha_1 \text{ and } \mathcal{M}, u \models \alpha_2; \\
\mathcal{M}, u \models \beta & \Longleftrightarrow & \mathcal{M}, u \models \beta_1 \text{ or } \mathcal{M}, u \models \beta_2; \\
\mathcal{M}, u \models \pi^r & \Longleftrightarrow & \mathcal{M}, w \models \pi^r_0 \text{ for some } w \in \mathcal{U} \text{ s.t. } \mathcal{R}_r(u, w) \text{ holds in } \mathcal{M}; \\
\mathcal{M}, u \models \nu^r & \Longleftrightarrow & \mathcal{M}, w \models \nu^r_0 \text{ for every } w \in \mathcal{U} \text{ s.t. } \mathcal{R}_r(u, w) \text{ holds in } \mathcal{M}.
\end{array}
$$

"$\mathcal{M}, u \models \varphi$" should be read as "$\mathcal{M}, u$ *satisfy* φ in K_m" (alternatively, "$\mathcal{M}, u$ K_m-satisfies φ"). We say that a K_m-formula φ is satisfiable in K_m (K_m-satisfiable from now on) if

[1] Available at `http://www.dit.unitn.it/~rseba/sat06/extended.ps`

and only if there exist $\mathcal{M}$ and $u \in M$ s.t. $\mathcal{M}, u \models \varphi$. (When this causes no ambiguity, we sometimes drop the prefix "K_m-".) We say that w is a *successor* of u through $\mathcal{R}_r$ iff $\mathcal{R}_r(u,w)$ holds in $\mathcal{M}$.

The problem of determining the K_m-satisfiability of a K_m-formula φ is decidable and PSPACE-complete [9, 6], even restricting the language to a single boolean atom (i.e., $\mathcal{A} = \{A_1\}$) [5]; if we impose a bound on the modal depth of the K_m-formulas, the problem reduces to NP-complete [5]. For a more detailed description on K_m— including, e.g., axiomatic characterization, decidability and complexity results — see [6, 5].

A K_m-formula is said to be in *Negative Normal Form (NNF)* if it is written in terms of the symbols $\Box_r$, $\Diamond_r$, $\wedge$, $\vee$ and propositional literals A_i, $\neg A_i$ (i.e., if all negations occur only before propositional atoms in $\mathcal{A}$). Every K_m-formula φ can be converted into an equivalent one $NNF(\varphi)$ by recursively applying the rewriting rules: $\neg\Box_r\varphi \Longrightarrow \Diamond_r\neg\varphi$, $\neg\Diamond_r\varphi \Longrightarrow \Box_r\neg\varphi$, $\neg(\varphi_1 \wedge \varphi_2) \Longrightarrow (\neg\varphi_1 \vee \neg\varphi_2)$, $\neg(\varphi_1 \vee \varphi_2) \Longrightarrow (\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg\neg\varphi \Longrightarrow \varphi$.

A K_m-formula is said to be in *Box Normal Form (BNF)* [11, 12] if it is written in terms of the symbols $\Box_r$, $\neg\Box_r$, $\wedge$, $\vee$, and propositional literals A_i, $\neg A_i$ (i.e., if no diamonds are there, and all negations occurs only before boxes or before propositional atoms in $\mathcal{A}$). Every K_m-formula φ can be converted into an equivalent one $BNF(\varphi)$ by recursively applying the rewriting rules: $\Diamond_r\varphi \Longrightarrow \neg\Box_r\neg\varphi$, $\neg(\varphi_1 \wedge \varphi_2) \Longrightarrow (\neg\varphi_1 \vee \neg\varphi_2)$, $\neg(\varphi_1 \vee \varphi_2) \Longrightarrow (\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg\neg\varphi \Longrightarrow \varphi$.

3 The Encoding

We borrow some notation from the *Single Step Tableau (SST)* framework [10]. We represent univocally states in $\mathcal{M}$ as labels σ, represented as non empty sequences of integers $1.n_1^{r_1}.n_2^{r_2}. \ldots .n_k^{r_k}$, s.t. the label 1 represents the root state, and $\sigma.n^r$ represents the n-th successor of σ through the relation $\mathcal{R}_r$.

Notationally, we often write "$(\bigwedge_i l_i) \rightarrow \bigvee_j l_j$" for the clause "$\bigvee_j \neg l_i \vee \bigvee_j l_j$", and "$(\bigwedge_i l_i) \rightarrow (\bigwedge_j l_j)$" for the conjunction of clauses "$\bigwedge_j(\bigvee_i \neg l_i \vee l_j)$".

3.1 The Basic Encoding

-Let $A_{[,\,]}$ be an *injective* function which maps a pair $\langle\sigma, \psi\rangle$, s.t. σ is a state label and ψ is a K_m-formula which is not in the form $\neg\phi$, into a boolean variable $A_{[\sigma,\ \psi]}$. Let $L_{[\sigma,\ \psi]}$ denote $\neg A_{[\sigma,\ \phi]}$ if ψ is in the form $\neg\phi$, $A_{[\sigma,\ \psi]}$ otherwise. Given a K_m-formula φ, the encoder K_m2SAT builds a boolean CNF formula as follows:

$$K_m2SAT(\varphi) := A_{[1,\ \varphi]} \wedge Def(1,\ \varphi), \qquad (1)$$

$$Def(\sigma,\ A_i), := \top \qquad (2)$$

$$Def(\sigma,\ \neg A_i) := \top \qquad (3)$$

$$Def(\sigma,\ \alpha) := (L_{[\sigma,\ \alpha]} \rightarrow (L_{[\sigma,\ \alpha_1]} \wedge L_{[\sigma,\ \alpha_2]})) \wedge Def(\sigma,\ \alpha_1) \wedge Def(\sigma,\ \alpha_2) \qquad (4)$$

$$Def(\sigma,\ \beta) := (L_{[\sigma,\ \beta]} \rightarrow (L_{[\sigma,\ \beta_1]} \vee L_{[\sigma,\ \beta_2]})) \wedge Def(\sigma,\ \beta_1) \wedge Def(\sigma,\ \beta_2) \qquad (5)$$

$$Def(\sigma,\ \pi^{r,j}) := (L_{[\sigma,\ \pi^{r,j}]} \rightarrow L_{[\sigma.j,\ \pi_0^{r,j}]}) \wedge Def(\sigma.j,\ \pi_0^{r,j}) \qquad (6)$$

$$Def(\sigma,\ \nu^r) := \bigwedge_{\langle\sigma:\pi^{r,i}\rangle} ((L_{[\sigma,\ \nu^r]} \wedge L_{[\sigma,\ \pi^{r,i}]}) \rightarrow L_{[\sigma.i,\ \nu_0^r]}) \wedge \bigwedge_{\langle\sigma:\pi^{r,i}\rangle} Def(\sigma.i,\ \nu_0^r). \qquad (7)$$

Here by "$\langle\sigma : \pi^{r,i}\rangle$" we mean that $\pi^{r,i}$ is the j-th dinstinct π^r formula labeled by σ.

We assume that the K_m-formulas are represented as DAGs, so that to avoid the expansion of the same $Def(\sigma,\ \psi)$ more than once. Moreover, following [10], we assume that, for each σ, the $Def(\sigma,\ \psi)$'s are expanded in the order: α, β, π, ν. Thus, each $Def(\sigma,\ \nu^r)$ is expanded after the expansion of all $Def(\sigma,\ \pi^{r,i})$'s, so that $Def(\sigma,\ \nu^r)$ will generate one clause $((L_{[\sigma,\ \pi^{r,i}]} \wedge L_{[\sigma,\ \Box_r \nu_0^r]}) \rightarrow L_{[\sigma.i,\ \nu_0^r]})$ and one novel definition $Def(\sigma.i,\ \nu_0^r)$ for each $Def(\sigma,\ \pi^{r,i})$ expanded.

Theorem 1. *A K_m-formula φ is K_m-satisfiable if and only if the corresponding boolean formula $K_m 2SAT(\varphi)$ is satisfiable.*

Notice that, due to (7), the number of variables and clauses in $K_m 2SAT(\varphi)$ may grow exponentially with $depth(\varphi)$. This is in accordance to what stated in [5].

3.2 Variants

Before the encoding, some potentially useful preprocessing can be performed.

First, the input K_m-formulas can be converted into NNF (like, e.g., in [10]) or into BNF (like, e.g., in [3, 11]). One potential advantage of the latter is that, when one $\Box_r \psi$ occurs both positively and negatively (like, e.g., in $(\Box_r \psi \vee \ldots) \wedge (\neg\Box_r \psi \vee \ldots) \wedge \ldots$), then both occurrences of $\Box_r \psi$ are labeled by the same boolean atom $A_{[\sigma,\ \Box_r \psi]}$, and hence they are always assigned the same truth value by DPLL; with NNF, instead, the negative occurrence $\neg\Box_r \psi$ is rewritten into $\Diamond_r \neg\psi$, so that two distinct boolean atoms $A_{[\sigma,\ \Box_r \psi]}$ and $A_{[\sigma,\ \Diamond_r \neg\psi]}$ are generated; DPLL can assign them the same truth value, creating a hidden conflict which may require some extra boolean search to reveal.

Example 1 (NNF).
Let φ_{nnf} be $(\Diamond A_1 \vee \Diamond(A_2 \vee A_3)) \ \wedge\ \Box\neg A_1 \ \wedge\ \Box\neg A_2 \ \wedge\ \Box\neg A_3$.[2] It is easy to see that φ_{nnf} is K_1-unsatisfiable. $K_m 2SAT(\varphi_{nnf})$ is:

1. $A_{[1,\ \varphi_{nnf}]}$
2. $\wedge\ \ (A_{[1,\ \varphi_{nnf}]} \rightarrow (A_{[1,\ \Diamond A_1 \vee \Diamond(A_2 \vee A_3)]} \wedge A_{[1,\ \Box\neg A_1]} \wedge A_{[1,\ \Box\neg A_2]} \wedge A_{[1,\ \Box\neg A_3]}))$
3. $\wedge\ \ (A_{[1,\ \Diamond A_1 \vee \Diamond(A_2 \vee A_3)]} \rightarrow (A_{[1,\ \Diamond A_1]} \vee A_{[1,\ \Diamond(A_2 \vee A_3)]}))$
4. $\wedge\ \ (A_{[1,\ \Diamond A_1]} \rightarrow A_{[1.1,\ A_1]})$
5. $\wedge\ \ (A_{[1,\ \Diamond(A_2 \vee A_3)]} \rightarrow A_{[1.2,\ A_2 \vee A_3]})$
6. $\wedge\ \ ((A_{[1,\ \Box\neg A_1]} \wedge A_{[1,\ \Diamond A_1]}) \rightarrow \neg A_{[1.1,\ A_1]})$
7. $\wedge\ \ ((A_{[1,\ \Box\neg A_2]} \wedge A_{[1,\ \Diamond A_1]}) \rightarrow \neg A_{[1.1,\ A_2]})$
8. $\wedge\ \ ((A_{[1,\ \Box\neg A_3]} \wedge A_{[1,\ \Diamond A_1]}) \rightarrow \neg A_{[1.1,\ A_3]})$
9. $\wedge\ \ ((A_{[1,\ \Box\neg A_1]} \wedge A_{[1,\ \Diamond(A_2 \vee A_3)]}) \rightarrow \neg A_{[1.2,\ A_1]})$
10. $\wedge\ \ ((A_{[1,\ \Box\neg A_2]} \wedge A_{[1,\ \Diamond(A_2 \vee A_3)]}) \rightarrow \neg A_{[1.2,\ A_2]})$
11. $\wedge\ \ ((A_{[1,\ \Box\neg A_3]} \wedge A_{[1,\ \Diamond(A_2 \vee A_3)]}) \rightarrow \neg A_{[1.2,\ A_3]})$
12. $\wedge\ \ (A_{[1.2,\ A_2 \vee A_3]} \rightarrow (A_{[1.2,\ A_2]} \vee A_{[1.2,\ A_3]}))$

After a run of BCP, 3. reduces to the implicate disjunction. If the first element $A_{[1,\ \Diamond A_1]}$ is assigned, then by BCP we have a conflict on 4.,6. If the second element $A_{[1,\ \Diamond(A_2 \vee A_3)]}$ is assigned, then by BCP we have a conflict on 12. Thus $K_m 2SAT(\varphi_{nnf})$ is unsatisfiable.
$\Diamond$

[2] For K_1 formulas, we omit the box and diamond indexes.

Example 2 (BNF).
Let $\varphi_{bnf} = (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3)) \wedge \Box\neg A_1 \wedge \Box\neg A_2 \wedge \Box\neg A_3$. It is easy to see that φ_{bnf} is K_1-unsatisfiable. $K_m 2SAT(\varphi_{bnf})$ is:

1. $\quad A_{[1,\ \varphi_{bnf}]}$
2. $\wedge\ (A_{[1,\ \varphi_{bnf}]} \rightarrow (A_{[1,\ (\neg\Box\neg A_1\vee\neg\Box(\neg A_2\wedge\neg A_3))]} \wedge A_{[1,\ \Box\neg A_1]} \wedge A_{[1,\ \Box\neg A_2]} \wedge A_{[1,\ \Box\neg A_3]}))$
3. $\wedge\ (A_{[1,\ (\neg\Box\neg A_1\vee\neg\Box(\neg A_2\wedge\neg A_3))]} \rightarrow (\neg A_{[1,\ \Box\neg A_1]} \vee \neg A_{[1,\ \Box(\neg A_2\wedge\neg A_3)]}))$
4. $\wedge\ (\neg A_{[1,\ \Box\neg A_1]} \rightarrow A_{[1.1,\ A_1]})$
5. $\wedge\ (\neg A_{[1,\ \Box(\neg A_2\wedge\neg A_3)]} \rightarrow \neg A_{[1.2,\ (\neg A_2\wedge\neg A_3)]})$
6. $\wedge\ ((A_{[1,\ \Box\neg A_1]} \wedge \neg A_{[1,\ \Box\neg A_1]}) \rightarrow \neg A_{[1.1,\ A_1]})$
7. $\wedge\ ((A_{[1,\ \Box\neg A_2]} \wedge \neg A_{[1,\ \Box\neg A_1]}) \rightarrow \neg A_{[1.1,\ A_2]})$
8. $\wedge\ ((A_{[1,\ \Box\neg A_3]} \wedge \neg A_{[1,\ \Box\neg A_1]}) \rightarrow \neg A_{[1.1,\ A_3]})$
9. $\wedge\ ((A_{[1,\ \Box\neg A_1]} \wedge \neg A_{[1,\ \Box(\neg A_2\wedge\neg A_3)]}) \rightarrow \neg A_{[1.2,\ A_1]})$
10. $\wedge\ ((A_{[1,\ \Box\neg A_2]} \wedge \neg A_{[1,\ \Box(\neg A_2\wedge\neg A_3)]}) \rightarrow \neg A_{[1.2,\ A_2]})$
11. $\wedge\ ((A_{[1,\ \Box\neg A_3]} \wedge \neg A_{[1,\ \Box(\neg A_2\wedge\neg A_3)]}) \rightarrow \neg A_{[1.2,\ A_3]})$
12. $\wedge\ (\neg A_{[1.2,\ (\neg A_2\wedge\neg A_3)]} \rightarrow (A_{[1.2,\ A_2]} \vee A_{[1.2,\ A_3]}))$

Unlike with NNF, $K_m 2SAT(\varphi_{bnf})$ is found unsatisfiable directly by BCP. Notice that the unit-propagation of $A_{[1,\ \Box\neg A_1]}$ from 2. causes $\neg A_{[1,\ \Box\neg A_1]}$ in 3. to be false, so that one of the two (unsatisfiable) branches induced by the disjunction is cut a priori. With NNF, the corresponding atoms $A_{[1,\ \Box\neg A_1]}$ and $A_{[1,\ \Diamond A_1]}$ are not recognized to be one the negation of the other, s.t. DPLL may need exploring one boolean branch more. $\diamond$

Second, the (NNF or BNF) K_m-formula can also be rewritten by recursively applying the validity-preserving "box/diamond lifting rules":

$$(\Box_r\varphi_1 \wedge \Box_r\varphi_2) \Longrightarrow \Box_r(\varphi_1 \wedge \varphi_2), \quad (\Diamond_r\varphi_1 \vee \Diamond_r\varphi_2) \Longrightarrow \Diamond_r(\varphi_1 \vee \varphi_2). \tag{8}$$

This has the potential benefit of reducing the number of $\pi^{r,i}$ formulas, and hence the number of labels $\sigma.i$ to take into account in the expansion of the $Def(\sigma,\ \nu^r)$'s.

Example 3 (BNF with LIFT).
Let $\varphi_{bnflift} = \neg\Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3) \wedge \Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3)$. It is easy to see that $\varphi_{bnflift}$ is K_1-unsatisfiable. $K_m 2SAT(\varphi_{bnflift})$ is:

1. $\quad A_{[1,\ \varphi_{bnflift}]}$
2. $\wedge\ (A_{[1,\ \varphi_{bnflift}]} \rightarrow (\neg A_{[1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)]} \wedge A_{[1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)]}))$
3. $\wedge\ (\neg A_{[1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)]} \rightarrow \neg A_{[1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)]})$
4. $\wedge\ (\neg A_{[1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)]} \rightarrow (A_{[1.1,\ A_1]} \vee A_{[1.1,\ A_2]} \vee A_{[1.1,\ A_3]}))$

$K_m 2SAT(\varphi_{bnflift})$ is found unsatisfiable by BCP. $\diamond$

One potential drawback of applying the lifting rules is that, by collapsing $(\Box_r\varphi_1 \wedge \Box_r\varphi_2)$ into $\Box_r(\varphi_1 \wedge \varphi_2)$ and $(\Diamond_r\varphi_1 \vee \Diamond_r\varphi_2)$ into $\Diamond_r(\varphi_1 \vee \varphi_2)$, the possibility of sharing box/diamond subformulas in the DAG representation of φ is reduced.

4 Conclusions and future work

In this paper (see also the extended version) we have explored the idea of encoding $K_m/\mathcal{ALC}$-satisfiability into SAT, so that to be handled by state-of-the-art SAT tools. We have showed that, despite the intrinsic risk of blowup in the size of the encoded formulas, the performances of this approach are comparable with those of current state-of-the-art tools on a rather extensive variety of empirical tests. (Notice that, as a byproduct of this work, the encoding of hard K_m-formulas could be used as benchmarks for SAT solvers.)

We see many possible direction to explore in order to enhance and extend this approach. First, our current implementation of the encoder is very straightforward, and optimizations for making the formula more compact can be introduced. Second, techniques implemented in other approaches (e.g., the pure literal optimization of [12]) could be imported. Third, hybrid approaches between K_m2SAT and KSAT-style tools could be investigated.

Another important open research line is to explore encodings for other modal and description logics. Whilst for logics like T_m the extension should be straightforward, logics like $S4_m$, or more elaborated description logics than $\mathcal{ALC}$, should be challenging.

References

1. S. Brand, R. Gennari, and M. de Rijke. Constraint Programming for Modelling and Solving Modal Satisfability. In *Proc. CP 2003*, volume 3010 of *LNAI*. Springer, 2003.
2. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishg, 1983.
3. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. CADE'13*. Springer, 1996.
4. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). *Information and Computation*, 162(1/2), 2000.
5. J. 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(3):361–372, 1995.
6. J.Y. Halpern and Y. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.
7. I. Horrocks and P. F. Patel-Schneider. Optimizing Description Logic Subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
8. U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption Testing with SPASS. In *Proc. DL'99*, pages 136–137, 1999.
9. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comp.*, 6(3):467–480, 1977.
10. F. Massacci. Single Step Tableaux for modal logics: methodology, computations, algorithms. *Journal of Automated Reasoning*, Vol. 24(3), 2000.
11. G. Pan, U. Sattler, and M. Y. Vardi. BDD-Based Decision Procedures for K. In *Proc. CADE*, LNAI. Springer, 2002.
12. G. Pan and M. Y. Vardi. Optimizing a BDD-based modal solver. In *Proc. CADE*, LNAI. Springer, 2003.
13. R. Sebastiani and M. Vescovi. Encoding the satisfiability of modal and description logics into SAT: the case study of K(m)/$\mathcal{ALC}$. Extended version. Technical Report DIT-06-033, DIT, University of Trento., May 2006. Available at `http://www.dit.unitn.it/~rseba/sat06/extended.ps`.