Solving the Satisfiability Problem of Modal Logic S5 Guided by Graph Coloring

Pei Huang[1,3], **Minghao Liu**[1,3], **Ping Wang**[1,3], **Wenhui Zhang**[1,3], **Feifei Ma**[1,2,3*], **Jian Zhang**[1,3*]

[1]State Key Laboratory of Computer Science, ISCAS, China
[2]Laboratory of Parallel Software and Computational Science, ISCAS, China
[3]University of Chinese Academy of Sciences
{huangpei, liumh, wangping, zwh, maff, zj}@ios.ac.cn

Abstract

Modal logic S5 has found various applications in artificial intelligence. With the advances in modern SAT solvers, SAT-based approach has shown great potential in solving the satisfiability problem of S5. The scale of the SAT encoding for S5 is strongly influenced by the upper bound on the number of possible worlds. In this paper, we present a novel SAT-based approach for S5 satisfiability problem. We show a normal form for S5 formulas. Based on this normal form, a conflict graph can be derived whose chromatic number provides an upper bound of the possible worlds and a lot of unnecessary search spaces can be eliminated in this process. A heuristic graph coloring algorithm is adopted to balance the efficiency and optimality. The number of possible worlds can be significantly reduced for many practical instances. Extensive experiments demonstrate that our approach outperforms state-of-the-art S5-SAT solvers.

1 Introduction

Modal logic provides a theoretical framework for important applications in many areas of artificial intelligence, including game theory [Lorini and Schwarzentruber, 2010], knowledge compilation [Bienvenu *et al.*, 2010], contingent planning [Niveau and Zanuttini, 2016] and formal verification [Aguilera and Fernández-Duque, 2016; Fairtlough and Mendler, 1994]. In the past decades, automated reasoning techniques for modal logic has been vastly studied (e.g. [Balco *et al.*, 2018; Giunchiglia and Sebastiani, 1996; Kaminski and Tebbi, 2013]).

S5 is a well-known modal logic system, which is suitable for representing and reasoning about the knowledge of a single agent [Fagin *et al.*, 2004]. Recently, modal logic S5 is used in knowledge compilation [Bienvenu *et al.*, 2010; Niveau and Zanuttini, 2016] and epistemic planner [Wan *et al.*, 2015]. So, it promotes us to develop and improve automated reasoning technique for modal logic S5.

There are four common ways to tackle the modal logic satisfiability problems: tableau methods [Götzmann *et al.*, 2010; Gasquet *et al.*, 2005], first-order logic based methods [Ohlbach, 1991], resolution methods [Auffray *et al.*, 1990; Nalon *et al.*, 2017] and SAT-based methods [Kaminski and Tebbi, 2013; Lagniez *et al.*, 2018]. Due to the improvement of modern SAT solvers, SAT-based methods are showing their potential and strength such as the-state-of-the-art S5 solver named *S52SAT* [Caridroit *et al.*, 2017]. The SAT-based method has three advantages: 1) It can easily return a model (compared with resolution methods). 2) It can learn from the conflicts (CDCL) and utilize them. 3) It makes a good balance between guess and reasoning.

S5-SAT can be reduced to SAT based on the theoretical proof showed in [Ladner, 1977; Fagin *et al.*, 2004]. However, naive SAT translation method will produce numerous variables and clauses. For some relatively large S5 formulas, it will cause hundreds of thousands of variables and millions of clauses. Existing SAT-based methods frequently run out of memory for these intractable scale formulas. Besides, a large number of variables and clauses will wear away the efficiency and the advantages of the SAT core.

In this paper, we propose a novel SAT-based method for S5-SAT which can save a lot of memory and significantly improve the computation efficiency. We first show a CNF-like normal form and name it S5-NF. Based on this normal form, a diamond conflict graph can be derived whose chromatic number provides an upper bound on the number of necessary worlds. In practice, the chromatic number is obtained approximately with a heuristic graph coloring algorithm, so that the extra cost is quite limited. Moreover, a lot of unnecessary search spaces can be eliminated in this process. Experimental results show that our solver outperforms the state-of-the-art S5 solvers, on some instances even by orders of magnitudes in efficiency. Besides, our solver consumes much less memory on many large scale S5 formulas.

This paper is organized as follows: first, we introduce some preliminaries about modal logic S5; then we detail the S5-NF; after that, the diamond conflict checking (DCC) strategy and the framework of solving S5-SAT will be expounded; furthermore, we evaluate and discuss the experimental results; in the final section, conclusions are drawn.

2 Preliminaries

This section briefly reviews the syntax and the semantics of modal logic S5. The set of formulas ϕ of S5 is a language

*Corresponding Authors

$\mathcal{L}$ which extends the propositional language with the modal connectives (or modal operators) $\Box$ and $\Diamond$. $\Box$ (box) means ***necessity*** and $\Diamond$ (diamond) means ***possibility***. For example, $\Box p$ expresses the concept that "It is necessary that p" and $\Diamond p$ expresses the concept that "It is possible that p". The language is defined by the grammar:

$$\phi ::= \bot \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \Box\phi \mid \Diamond\phi$$

where $p \in \mathbb{P}$ and $\mathbb{P}$ denotes a countably infinite non-empty set of propositional variables. Logical connectives '$\rightarrow$' and '$\leftrightarrow$' are omitted here.

Now-standard Kripke semantics for modal logic defines a frame, which consists of a non-empty set W, and a binary relation R. The members of W are generally called possible worlds. The relation R, also known as the **accessibility relation**, is defined between the possible worlds in W. For example, $w\ R\ w'$ means the world w' is accessible from world w. I is a function $W \times \mathbb{P} \rightarrow \{0, 1\}$.

S5 has the following axioms:

$\mathcal{K}$. $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

$\mathcal{T}$. $\Box A \rightarrow A$

$\mathcal{B}$. $A \rightarrow \Box\Diamond A$

4. $\Box A \rightarrow \Box\Box A$

These axioms imply that the relation R in S5 is **reflexive**, **symmetric** and **transitive**. So the possible-worlds semantics for S5 can be simplified as a simple version without accessibility relation [Fitting, 1999]. The satisfiability relation $\vDash$ for formulas in $\mathcal{L}$ is recursively defined as follows:

$(W, I, w) \vDash \top$

$(W, I, w) \vDash p$ iff $I(w, p) = 1$

$(W, I, w) \vDash \neg\phi$ iff $(W, I, w) \nvDash \phi$

$(W, I, w) \vDash \phi \wedge \varphi$ iff $(W, I, w) \vDash \phi$ and $(W, I, w) \vDash \varphi$

$(W, I, w) \vDash \phi \vee \varphi$ iff $(W, I, w) \vDash \phi$ or $(W, I, w) \vDash \varphi$

$(W, I, w) \vDash \Box\phi$ iff $\forall w' \in W, (W, I, w') \vDash \phi$

$(W, I, w) \vDash \Diamond\phi$ iff $\exists w' \in W, (W, I, w') \vDash \phi$

Following the semantics (satisfiability relation), we have a list of equivalences as follows.

i) $\Box\top \leftrightarrow \top$, $\Box\bot \leftrightarrow \bot$, $\Diamond\top \leftrightarrow \top$, $\Diamond\bot \leftrightarrow \bot$

ii) $\neg\Diamond\phi \leftrightarrow \Box\neg\phi$, $\neg\Box\phi \leftrightarrow \Diamond\neg\phi$

iii) $\Box(\phi \wedge \varphi) \leftrightarrow \Box\phi \wedge \Box\varphi$

iv) $\Diamond(\phi \vee \varphi) \leftrightarrow \Diamond\phi \vee \Diamond\varphi$

The diamond degree of a negation normal form ϕ, $dd(\phi)$, is defined recursively as:

$dd(\top) = dd(\neg\top) = dd(p) = dd(\neg p) = 0$

$dd(\phi \wedge \varphi) = dd(\phi) + dd(\varphi)$

$dd(\phi \vee \varphi) = max(dd(\phi), dd(\varphi))$

$dd(\Box\phi) = dd(\phi)$

$dd(\Diamond\phi) = 1 + dd(\phi)$

3 A Normal Form for S5

A normal form for S5 formulas, called S5-NF, is defined in this section. It is a kind of CNF-like normal form. Actually, many CNF-like normal forms were defined in previous works for resolution such as [del Cerro, 1982; Enjalbert and del Cerro, 1989; Salhi and Sioutis, 2015]. The S5-NF is similar to these CNF-like normal forms but with some textural difference. It is designed for our solving framework, and we will show how to transform an S5 formula into an S5-NF.

3.1 Reduction Rules

First, some key rules in the transformation process are listed as follows. Note that $\odot \in \{\Box, \Diamond\}$.

1) $\odot^n\Box\phi \Rightarrow \Box\phi$

2) $\odot^n\Diamond\phi \Rightarrow \Diamond\phi$

3) $\Box(\phi_1 \wedge \phi_2 \wedge ... \wedge \phi_n) \Rightarrow \Box\phi_1 \wedge \Box\phi_2 \wedge ... \wedge \Box\phi_n$

4) $\Diamond(\phi_1 \vee \phi_2 \vee ... \vee \phi_n) \Rightarrow \Diamond\phi_1 \vee \Diamond\phi_2 \vee ... \vee \Diamond\phi_n$

5) $\Box(\varphi_1 \vee \varphi_2 \vee ... \vee \varphi_m \vee \odot\phi_1 \vee \odot\phi_2 \vee ... \vee \odot\phi_n)$
$\Rightarrow \Box(\varphi_1 \vee \varphi_2 \vee ... \vee \varphi_m) \vee \odot\phi_1 \vee \odot\phi_2 \vee ... \vee \odot\phi_n$

6) $\Diamond(\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_m \wedge \odot\phi_1 \wedge \odot\phi_2 \wedge ... \wedge \odot\phi_n)$
$\Rightarrow \Diamond(\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_m) \wedge \odot\phi_1 \wedge \odot\phi_2 \wedge ... \wedge \odot\phi_n$

The correctness of these rules is easy to verify. We present the sketch of the proof of these rules.

Lemma 1. *$\Diamond\Box p \leftrightarrow \Box p$, $\Box\Box p \leftrightarrow \Box p$, $\Box\Diamond p \leftrightarrow \Diamond p$, $\Diamond\Diamond p \leftrightarrow \Diamond p$.*

Lemma 1 can be easily proved according to the reflexivity, symmetry and transitivity of S5.

Applying these rules recursively we can conclude that 1) and 2) hold. These two rules can simplify multiple modal operators to one, such as $\Box\Diamond\Box\Diamond\Box\Box\phi \leftrightarrow \Box\phi$. That means each formula can only keep the nearest modal operator.

Rules 3) and 4) can be easily proved via iii) and iv) of the equivalences listed previously.

Rules 5) and 6) can be seen as supplements of 3) and 4). In fact, we can not rewrite $\Box(p \vee q)$ to the disjunction $\Box p \vee \Box q$ because $\Box(p \vee q) \nleftrightarrow \Box p \vee \Box q$. But we can rewrite $\Box(p \vee \Diamond q)$ to $\Box p \vee \Diamond q$, as formulated in the following lemma.

Lemma 2. *$\Box(p \vee \Diamond q) \leftrightarrow \Box p \vee \Diamond q$, $\Box(p \vee \Box q) \leftrightarrow \Box p \vee \Box q$, $\Diamond(p \wedge \Box q) \leftrightarrow \Diamond p \wedge \Box q$, $\Diamond(p \wedge \Diamond q) \leftrightarrow \Diamond p \wedge \Diamond q$*

One can easily prove the lemma via axioms in S5. Applying these rules recursively we can prove that 5) and 6) hold.

3.2 S5 Normal Form

S5-NF is similar to the CNF in propositional logic. So, we extend some concepts from propositional logic to name the components of S5-NF. Any S5 formula can be reduced to an equivalent one of modal degree 0 or 1 [Mints and Minc, 1992]. S5-NF is a kind of first degree normal form.

Definition 1 (S5-literal)**.** *Propositional literal p, $\Box(p_1 \vee p_2 \vee ... \vee p_j)$ and $\Diamond(p_1 \wedge p_2 \wedge ... \wedge p_k)$ are called S5-literal, iff $\forall i \in \mathbb{N}$, p_i is a propositional literal. The S5-literal $\Diamond(p_1 \wedge p_2 \wedge ... \wedge p_k)$ is called **D-literal**, and $\Box(p_1 \vee p_2 \vee ... \vee p_j)$ is called **B-literal**.*

Example 1. *p, $\neg q$, $\Box p$, $\Diamond\neg p$, $\Box(\neg p \vee q \vee \neg r)$ and $\Diamond(p \wedge \neg q)$ are S5-literals.*

Definition 2 (S5-clause)**.** *If $\forall i \in \mathbb{N}$, l_i is an S5-literal then the disjunction of S5-literals, $(l_1 \vee l_2 \vee ... \vee l_n)$, is called S5-clause.*

Example 2. *$\Diamond\neg p \vee \Box(\neg p \vee q \vee \neg r) \vee \Diamond(p \wedge \neg q)$ is an S5-clause.*

Definition 3 (S5-NF)**.** *If $\forall i \in \mathbb{N}^+$, C_i is an S5-clause then the conjunction of S5-clauses, $C_1 \wedge C_2 \wedge ... \wedge C_n$, is called S5 normal form (S5-NF).*

Theorem 1. *Any S5 formula can be transformed into an equisatisfiable S5-NF formula.*

Proof. Step1. Eliminate $\rightarrow$ and $\leftrightarrow$. Then, transform S5 formula into an equivalent negation normal form(NNF), that is, negations appear only in front of propositional variables (definition in [Robinson and Voronkov, 2001]).
Step2. Apply the 6 reduction rules and distributive rules recursively, until the formula only contains S5-literals and $\vee$, $\wedge$.
Step3. Transform the formula into conjunction normal form, in a way such as [Tseitin, 1968]. Note that during this step, each S5-literal remains intact.
After the above 3 steps, the formula is transformed into the S5-NF formula. □

Fig 1 shows an example that how to transform an S5 formula into an equivalent S5-NF formula. After transformation, we get six S5-clauses.

$$\Diamond\Box\Diamond\neg\{\Diamond(\neg p \wedge \neg q) \vee \Diamond(p \wedge q)\} \wedge \Diamond\{\Box(\neg p \vee r \vee s) \wedge p \wedge \neg q\} \wedge \{\neg r \rightarrow [\neg\Diamond(p \wedge q) \rightarrow \Diamond(p \wedge \neg q)]\} \wedge \Box\{\neg r \vee \Box\Diamond p\}$$

$$\Downarrow \text{Step1}$$

$$\Diamond\Box\Diamond\{\Box(p \vee q) \wedge \Box(\neg p \vee \neg q)\} \wedge \Diamond\{\Box(\neg p \vee r \vee s) \wedge p \wedge \neg q\} \wedge \{r \vee \Diamond(p \wedge q) \vee \Diamond(p \wedge \neg q)\} \wedge \Box\{\neg r \vee \Box\Diamond p\}$$

$$\Downarrow \text{Step2, 3}$$

$$\underbrace{\Box(p \vee q)}_{C_1} \wedge \underbrace{\Box(\neg p \vee \neg q)}_{C_2} \wedge \underbrace{\Box(\neg p \vee r \vee s)}_{C_3} \wedge \underbrace{\Diamond(p \wedge \neg q)}_{C_4} \wedge \underbrace{\{r \vee \Diamond(p \wedge q) \vee \Diamond(p \wedge \neg q)\}}_{C_5} \wedge \underbrace{\{\Box\neg r \vee \Diamond p\}}_{C_6}$$

Figure 1: An example of transforming S5-NF

In step 2, We use distributive rules to show the feasibility. Theoretically, only using distributive rules can blow up formula size in some cases. Introducing new variables and applying a Tseitin-like transformation locally can alleviate this problem. The following equivalences can be used to encode subformulas and x is a new variable.

$$\Box(x \leftrightarrow p \vee q) \Leftrightarrow \Box(\neg x \vee p \vee q) \wedge \Box(x \vee \neg p) \wedge \Box(x \vee \neg q)$$
$$\Box(x \leftrightarrow p \wedge q) \Leftrightarrow \Box(x \vee \neg p \vee \neg q) \wedge \Box(\neg x \vee p) \wedge \Box(\neg x \vee q)$$

Example 3. *Consider the formula:*

$$\Diamond(\bigwedge_{i=1}^{N}(l_i^1 \vee l_i^2)) \wedge \Box(\bigvee_{j=1}^{M}(l_j^3 \wedge l_j^4))$$

It can be rewritten as a conjunction of the following two formulas with new variables x_i, y_j:

$$\Diamond(\bigwedge_{i=1}^{N} x_i) \wedge (\bigwedge_{i=1}^{N} \Box(x_i \leftrightarrow (l_i^1 \vee l_i^2)))$$
$$\Box(\bigvee_{j=1}^{M} y_j) \wedge (\bigwedge_{j=1}^{M} \Box(y_j \leftrightarrow (l_j^3 \wedge l_j^4)))$$

In practice, many problems are formalized as:

$$(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \rightarrow C$$

A_i is an axiom (assumption, knowledge, state) and C is a conjecture (goal). The task is to check whether the formula is a tautology. It is equivalent to proving the following one is unsatisfiable.

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n \wedge \neg C$$

In order to get S5-NF, we only need to transform each subformula A_i and $\neg C$ to S5-clause. Generally, the size of A_i and C are not too large.

4 Dimond Elimination

In this section, we introduce a basic method to eliminate the diamond operator based on S5-NF. This method is a variant of Skolemization. In first-order logic, Skolemization is a key technique in satisfiability testing which can remove existential quantifiers from formal logic statements. For example, $\exists x A(x)$ may be changed to $A(c)$, where c is a new constant. The difference is that our method will not introduce new functions. In order to make our method more readable, we use first-order logical language as a transition. How to model S5-NF with propositional logic language will be discussed in Section 6.

Definition 4. *Translation function $tr^-(\phi)$ for an S5-NF ϕ is a substitution procedure which can produce a new formula:*

1. $\top \Rightarrow \top \quad \bot \Rightarrow \bot$
2. *For all propositional literals: $p \Rightarrow p(0)$*
3. *For all B-literals:*
   $\Box(p \vee q \vee \cdots \vee s) \Rightarrow \forall x(p(x) \vee q(x) \vee \cdots \vee s(x))$
4. *For all D-literals:*
   $\Diamond(p \wedge q \wedge \cdots \wedge r) \Rightarrow (p(i) \wedge q(i) \wedge \cdots \wedge r(i))$
   The constant i is the index of the S5-clause C_i where $\Diamond(p \wedge q \wedge \cdots \wedge r) \in C_i$.

Example 4. *Based on the definition of $tr^-(\phi)$, the S5-NF in Figure 1 can be rewritten as:*
$\forall x(p(x) \vee q(x)) \wedge \forall x(\neg p(x) \vee \neg q(x)) \wedge \forall x(\neg p(x) \vee r(x) \vee s(x)) \wedge (p(4) \wedge \neg q(4)) \wedge \{r(0) \vee (p(5) \wedge q(5)) \vee (p(5) \wedge \neg q(5))\} \wedge (\forall x \neg r(x) \vee p(6))$

Theorem 2. *The formula $tr^-(\phi)$ is equisatisfiable with the S5-NF ϕ.*

Proof. i) Assume $tr^-(\phi)$ is satisfiable. That is, there exists a model $M = \langle D, I' \rangle$ for $tr^-(\phi)$ where D is a non-empty domain and I' is an interpretation function. Then we can construct a Kripke structure to satisfy ϕ where $W = D$ and $I(w_i, p) = 1$ iff $\langle i \rangle \in I'(p)$.

ii) Assume $\phi = \{C_1 \wedge C_2, ... \wedge C_n\}$ is satisfiable, then for each clause in ϕ, there exists at least one satisfied S5-literal. We pick out only one satisfied S5-literal from each C_i, these S5-literals can be denoted as $F = \bigwedge_{\lambda=1}^{x} r_\lambda \wedge \bigwedge_{\lambda=1}^{y} \Box(b_{\lambda,1} \vee \cdots \vee b_{\lambda,j_i}) \wedge \bigwedge_{\lambda=1}^{z} \Diamond(d_{\lambda,1} \wedge \cdots \wedge d_{\lambda,m_\lambda})$ where $x+y+z = n$. We know that $F \vdash \phi$, so the Kripke structure satisfies F can also satisfy ϕ.

1) Create $z+1$ possible worlds. $w_0 \in W$ and if the D-literal $\Diamond(d_{\lambda,1} \wedge \cdots \wedge d_{\lambda,m_\lambda})$ is picked out from C_i then $w_i \in W$. Next, for all r_λ, $(W, I, w_0) \vDash r_\lambda$.

2) For all B-literals $\Box(b_{\lambda,1} \vee \cdots \vee b_{\lambda,j_i})$, $\forall w \in W$, $(W, I, w) \vDash b_{\lambda,1} \vee \cdots \vee b_{\lambda,j_i}$.

3) For all D-literals, if the $\Diamond(d_{\lambda,1} \wedge \cdots \wedge d_{\lambda,m_\lambda})$ is picked out from C_i then we make $(W, I, w_i) \vDash d_{\lambda,1} \wedge \cdots \wedge d_{\lambda,m_\lambda}$.

After 3 steps, a Kripke structure satisfying F has been constructed. Based on this Kripke structure we can build a model $M = \langle D, I' \rangle$ for $tr^-(\phi)$. D is composed by all the constants that occur in the formula $tr^-(\phi)$ and element 0. Then make $\langle i \rangle \in I'(p)$ iff $(W, I, w_i) \vdash p$ □

An S5-NF ϕ and $tr^-(\phi)$ are equisatisfiable but may have different models. Some models can be eliminated in the translations. Intuitively, predicates $p(i)$ can be seen as the truth value of p in the world w_i. The diamond elimination process binds each D-literal with a specific world. For example, in Fig 2, $M1$, $M2$ and $M3$ are tree models for the original formula. However, $M2$ is eliminated from $tr^-(\phi)$, because q must be true in the world w_2. $M3$ is an implied situation, $\neg r$

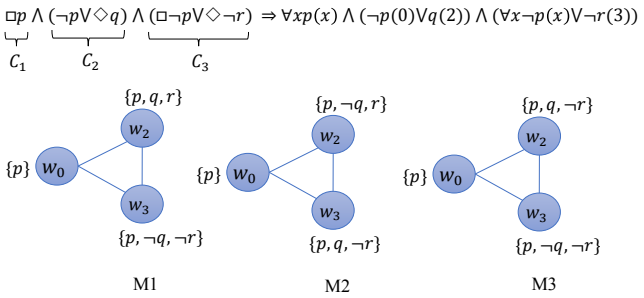

Figure 2: Three models of an S5-NF

It is easy to prove that if $tr^-(\phi)$ with m constants is satisfiable, then there exists a model M=$\langle D, I' \rangle$ satisfying the formula where $|D| = m$. So the satisfiability of the S5-NF ϕ can be decided with at most m possible worlds. Based on the definition of $dd(\phi)$, we know that $dd(\phi)$ is equivalent to $m-1$. It is consistent with the conclusion that the satisfiability of an S5 formula can be decided with at most $dd(\phi)+1$ possible worlds presented in [Caridroit *et al.*, 2017].

5 Diamond Conflict Checking (DCC)

In Sect. 4, we know that the satisfiability of an S5-NF ϕ can be decided with at most $dd(\phi)+1$ worlds. The D-literals in the same S5-clause can be realized via the same world and the D-literals in different S5-clauses are realized via different worlds. In this context, a question naturally arises: whether can some worlds be reused to realize the D-literals in different S5-clauses? Deciding the satisfiability of ϕ with $dd(\phi)+1$ worlds is in the sense of 'at the most' which considers the worst case. Sometimes, an S5 formula can be satisfied with a small-model where $|W|$ is smaller than the theoretical upper bound. For example, the S5-NF in Fig 2 can be satisfied with two possible worlds because $\Diamond q$ and $\Diamond \neg r$ can be realized via the same world simultaneously.

In this section, we propose a strategy called diamond conflict checking (DCC). The basic idea is to check whether D-literals in two S5-clauses can be in conflict when they are realized via the same world. Based on the conflict relation among these S5-clauses, we can build a graph. And, deciding which D-literals in different S5-clauses will be realized via the same world can be reduced to graph coloring by nature.

Although the conjunctive form of the D-literals provide convenience to identify the conflicts, determining the conflicts precisely is still difficult. We propose a conservative estimation of the conflicts among D-literals. What the method can guarantee is that the estimated D-literals without conflicts can be realized via the same world in safety.

Two D-literals dl_1 and dl_2 may interact with each other through B-literals. For example, $\Diamond p$, $\Diamond q$ and $\Box(\neg p \vee \neg q)$ occur in a formula simultaneously. If $\Box(\neg p \vee \neg q)$ must be true, then $\Diamond p$ and $\Diamond q$ are in conflict when we try to realize them with the same world.

Suppose that the S5-NF formula under consideration is given and BL is the set of all B-literals in the given formula. Let $PL(l)$ denote the set of all the propositional literals in an S5-literal l. An effect propagation set of a propositional literal p is defined as:

$$EP(p) = \bigcup_{bl \in BL \text{ and } \neg p \in PL(bl)} PL(bl) \setminus \{\neg p\}$$

Then the D-literal infection set $DI(dl)$ of a D-literal dl is defined as:

$$DI_0(dl) = PL(dl)$$
$$DI_{n+1}(dl) = DI_n(dl) \bigcup_{p \in DI_n(dl)} EP(p)$$
$$DI(dl) = DI_\infty(dl)$$

$DI(dl)$ is an estimation of which literals may become true in the world w when we try to realize the D-literal dl via the w. Apart from the literals in $PL(dl)$, other literals also can be true due to the B-literals.

Example 5. *In Fig 1, we assume that dl is $\Diamond p$ in C_6, then:*

$DI_0(dl) = PL(dl) = \{p\}$
$DI_1(dl) = \{p, \neg q, r, s\}$ *Based on* $\Box(\neg p \vee \neg q), \Box(\neg p \vee r \vee s)$
$DI(dl) = DI_2(dl) = \{p, \neg q, r, s\}$ *Convergence*

In the first iteration, when we assume p in $DI_0(dl)$ is true, we can deduce r and s may become true based on $\Box(\neg p \vee r \vee s)$. So we have $r, s \in DI_1(dl)$. Based on $\Box(\neg p \vee \neg q)$, we have $\neg q \in DI_1(dl)$

It is easy to know that the iteration of $DI(dl)$ will converge in $2 \times V_n$ steps in the worst case where V_n is the number of propositional variables.

Definition 5. *Let dl_1 and dl_2 be two D-literals. dl_1 and dl_2 are said to be in **D-literal conflict** with each other, if there exists p such that $p \in PL(dl_2)$ and $\neg p \in DI(dl_1)$.*

The D-literal conflict relation is symmetric w
dl_1 and dl_2. In fact, the following holds:

> There exists p such that $p \in PL(dl_2)$ an
> $DI(dl_1)$ iff there exists q such that $q \in$
> and $\neg q \in DI(dl_2)$.

The details of the proof can be found in link[1].

Definition 6. *Two S5-Clauses C_1, C_2 are said* ***mond conflict*** *with each other, iff $\exists dl_1 \in C_1$ ar such that dl_1 and dl_2 are in D-literal conflict.*

Definition 7. ***Diamond conflict graph*** *of an noted as an ordered pair $G = (V, E)$. V=$\{C$ NF and C_i has at least one D-literal$\}$ and E = C_i and C_j are in diamond conflict with each oth*

Based on the diamond conflict graph, a vertex cess can be introduced to decide which D-literal clauses can be realized via the same world. The ing same color can reuse the same world. If t number is χ, then the upper bound of the possi decide the satisfiability of the S5-NF is $\chi + 1$. We name this framework as diamond conflict checking (DCC).

Algorithm 1: Dimond Conflict Checking (DCC)

Input: An S5-NF formula ϕ
Output: $Color(C_i)$ and χ
1 Construct diamond conflict graph $G = (V, E)$ of ϕ ;
2 Apply coloring algorithm to G ;
3 $\forall C_i \in V$, record $Color(C_i)$;
4 $\chi \leftarrow$ The chromatic number ;
5 **return** $Color(C_i)$ and χ

$Color(C_i)$ is the color of the clause C_i and $Color(C_i) \in \mathbb{N}^+$. Finding the optimal χ is NP-hard. It may cost too much time when the diamond conflict graph is large. Thus, we use a heuristic coloring algorithm when we implement this framework. The χ is found by a greedy algorithm which considers the vertices in descending order according to their degrees and assigns to each vertex the smallest available color in this order. This heuristic is sometimes called the Welsh-Powell algorithm [Welsh and Powell, 1967].

Improved by DCC, the 4th rule in definition 4 can be modified as:

• For all D-literals:

$\Diamond(p \wedge q \wedge \cdots \wedge r) \Rightarrow (p(c) \wedge q(c) \wedge \cdots \wedge r(c))$

The constant $c = Color(C_i)$ where $\Diamond(p \wedge q \wedge \cdots \wedge r) \in C_i$.

Fig 3 shows the diamond conflict graph of the S5-NF in Fig 1. The graph can be colored with two chromatics.

We assume that $Green \rightarrow 1$ and $Yellow \rightarrow 2$. Based on the coloring result in Fig 3, the S5-NF in Fig 1 can be rewritten as:

$\forall x(p(x) \vee q(x)) \wedge \forall x(\neg p(x) \vee \neg q(x)) \wedge \forall x(\neg p(x) \vee r(x) \vee s(x)) \wedge (p(1) \wedge \neg q(1)) \wedge \{r(0) \vee (p(2) \wedge q(2)) \vee (p(2) \wedge \neg q(2))\} \wedge (\forall x \neg r(x) \vee p(1))$

[1]http://www.square16.org/tools/s5cheetah

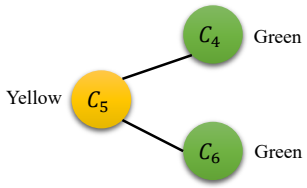

Figure 3: Diamond conflict graph

Compared with the basic method in Example 4, it saves a world to decide the satisfiability of the formula.

6 Solving Framework

In section 4 and 5 we use FOL and $tr^-(\phi)$ as medium to introduce our method. In this section, $tr^-(\phi)$ embedding with DCC will be modified to SAT version.

If an S5 formula with m modal operators is satisfiable, then there exists an S5-model satisfying the formula with at most $m+1$ worlds (upper-bound) [Ladner, 1977]. Since S5-SAT is NP-Complete, encoding the S5-SAT problem into SAT can be done in polynomial time. [Caridroit *et al.*, 2017] has shown a translation method. They also improve the upper-bound with the new metric called diamond degree. These two upper-bounds are theoretical results which consider 'the worst' case.

However, for many practical instances, the satisfiability of the formula can be decided with a small number of worlds. Our method can automatically discover that potential of the input S5 formulas. The upper-bound of necessary possible worlds ($\chi + 1$) is determined by a graph coloring algorithm.

Definition 8. *The translation function $tr(\phi)$ can produce a propositional formula for an input S5-NF ϕ with χ+1 worlds:*

1. $\top \Rightarrow \top \quad \bot \Rightarrow \bot$
2. *For all propositional literals:* $p \Rightarrow p_0$
3. *For all B-literals:*
   $\Box(p \vee q \vee \cdots \vee s) \Rightarrow \bigwedge_{j=0}^{\chi}(p_j \vee q_j \vee \cdots \vee s_j)$
4. *For all D-literals:*
   $\Diamond(p \wedge q \wedge \cdots \wedge r) \Rightarrow (p_c \wedge q_c \wedge \cdots \wedge r_c)$
   The constant c=$Color(C_i)$ where $\Diamond(p \wedge q \wedge \cdots \wedge r) \in C_i$.

Fresh Boolean variables p_i are added to the formula, denoting the truth value of p in the world w_i. In this function, the i represents the index of world. If two S5-clauses C_1 and C_2 are in different colors and both have $\Diamond\varphi$ as a subformula, then the SAT encoding of $\Diamond\varphi$ in them are different such as $\varphi_{color(C_1)}$ and $\varphi_{Color(C_2)}$. So, the framework of solving S5-SAT problem is shown as Algorithm 2.

Algorithm 2: The Framework of solving S5-SAT

Input: An S5 formula θ
Output: The satisfiability of θ
1 Transform the formula θ into an equivalent S5-NF ϕ;
2 $(Color(C_i), \chi) \leftarrow$ DCC(ϕ);
3 $\Sigma \leftarrow tr(\phi)$;
4 **return** $result \leftarrow$ SAT-Solver(Σ);

This framework can be divided into two stages. The first stage uses axioms and basic rules to reduce the formula to S5-NF and applies DCC to determine the upper bound. It can help the second stage avoid some unnecessary searching spaces and reduce the upper bound of possible worlds. It also has potential to improve other methods to tackle S5-SAT.

The second stage is trying to assign some variables and find a model via SAT solvers. Clauses learning, non-chronological backtracking, and good decision heuristics of modern SAT solvers make this stage to be efficient.

For SAT-based method, the number of variables and clauses of the propositional logic are greatly influenced by the upper bound of possible worlds. The S5-NF transformation process will eliminate a lot of modal operators and DCC will reuse some necessary worlds.

Suppose the input S5 formula θ has m modal operators, $dd(\theta)$ is diamond degree and χ is the chromatic number of its diamond conflict graph. Generally, χ is less than $dd(\theta)$ and m. For example, if θ is the original formula in Figure 2, then:

$$\chi = 2, dd(\theta) = 7, m = 12.$$

If the input is an S5-NF ϕ, the situation $\chi = dd(\phi)$ occurs when the diamond conflict graph is a complete graph.

7 Experiments

Based on our method, we implemented an S5-SAT solver named *S5cheetah*[1]. We compared it with the state-of-the-art solver *S52SAT* and tableau method solver *LCKS5TabProver (LCK)* [Abate *et al.*, 2007]. *S52SAT* uses Glucose 4.0 as the back-end SAT solver. For fair comparison, we also choose Glucose 4.0 as back-end SAT solver and use the same InTo-HyLo format benchmarks. *LCK* is able to read that format via a simple transformation.

The well-established modal logic benchmarks include $QMLTP$[2], $3CNF$ [Patel-Schneider and Sebastiani, 2003], $QS5$, $MQBF_K$ [Massacci, 1999], $TANCS2000_K$ [Massacci and Donini, 2000] and $LWB_{K,KT,S4}$ [Balsiger *et al.*, 2000]. $QMLTP$ are designed for testing automated theorem proving systems for first-order modal logics. It contains 177 propositional benchmarks for S5 from different domain (e.g. planning, querying databases, natural language processing, general algebra). All the benchmarks in $QS5$ are satisfiable which are generated based on hard combinatorial design about quasigroup. Note that $MQBF_K$, $TANCS2000_K$ and $LWB_{K,KT,S4}$ are not designed for S5. However, the results on those benchmarks are still valuable. They share the same grammar and S5-SAT entails K, KT and S4-SAT. All the benchmarks can be downloaded from the link[1]

The experiments are performed on a server with Intel(R) Xeon(R) CPU(2.40GHz), Ubuntu 16.04 and 64G RAM. We set the best performance parameters for *S52SAT*.

Table 1 shows the comparison of efficiency on each instance family. We report the number of instances which are firstly solved by each solver. And the mean time of tackling each instance family is listed in the column '$T_{avg}(ms)$'. The time bound is set to 120s and memory bound is set to 16GB for $QMLTP$, LWB, $MQBF_K$ (qbfMS), and $TANCS2000_K$ (qbfS, qbfL) as most instances of these benchmarks are relatively easy to solve. However, many instances in $QS5$, $MQBF_K$ (qbfML) and $3CNF$ cannot be solved by *S52SAT* within 120s with 16GB RAM. So, we set the time bound to 300s and memory bound to 64GB for them.

[2]http://www.iltp.de/qmltp/

Ins (#total)	S5cheetah		S52SAT		LCK	
	$\#Win$	T_{avg}	$\#Win$	T_{avg}	$\#Win$	T_{avg}
QMLTP (177)	174	3.26	0	33.08	3	23873.18
QS5 (252)	252	4122.89	0	107378.17	0	293357.79
3CNF (1000)	953	1964.31	46	40932.57	0	300000.00
LWB_k (378)	319	3862.77	18	11046.25	25	36487.63
LWB_kt (378)	346	3827.13	0	4621.00	21	31513.81
LWB_s4 (378)	333	3899.80	0	5058.72	34	33251.75
qbfL (80)	80	19.87	0	109.65	0	120000.00
qbfMS (240)	240	4.54	0	210.86	0	120000.00
qbfS (240)	240	2.20	0	69.08	0	120000.00
qbfML (240)	240	5148.92	0	83833.55	0	300000.00

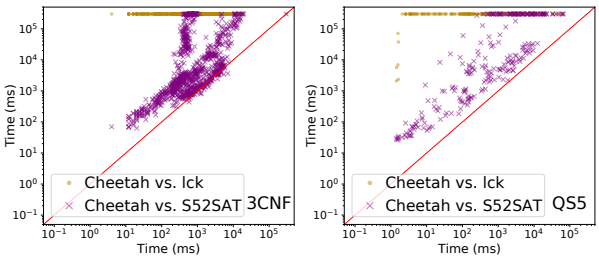

Figure 4: The comparison of run time on 3CNF and QS5.

In Table 1, the sum of #Win of all solvers is not necessary equal to #total, since there can be certain instances on which all solvers fail. We can see that our solver S5cheetah is more efficient on these benchmarks.

Figure 4 and 5 show the comparison of run time for all instances. The x-axis corresponds to the time used by S5Cheetah and the y-axis corresponds to the time used by *S52SAT* and *LCK*. The axes are in logarithmic scale. The points above the line y=x means that our solver consumes less time on these instances.

Most of the instances in 3CNF are unsatisfiable and all instances in QS5 are satisfiable. We solved all $QS5$ instances and 999 $3CNF$ instances within time bound while *S52SAT* solved 168 $QS5$ instances and 930 $3CNF$ instances.

Table 2 shows the average number of necessary possible worlds (#W), SAT variables (#Var) and clauses (#Clauses) of each instance family. We can see that our method reduces the number of necessary worlds. It generates fewer propositional variables and clauses compared with *S52SAT*.

Ins	S5cheetah			S52SAT		
	#W	#Var	#Clauses	#W	#Var	#Clauses
QMLTP	3.3	484.8	1386.15	7.5	941.1	3254.2
QS5	16.1	110365.3	278383.1	2826.7	3641914.2	12934423.3
3CNF	251.5	66395.9	1275616.9	329.1	391789.8	1909235.3
LWB_k	21.0	2039.3	71388.0	194.9	580469.9	1839919.7
LWB_kt	13.1	1877.6	8496.0	149.8	464861.9	1353167.9
LWB_s4	10.1	893.4	8054.5	218.2	568944.3	1807557.8
qbfL	18.1	3476.1	11766.5	22.5	16922.3	70339.1
qbfMS	10.5	36.7	793.1	96.9	19244.3	91272.4
qbfS	3.0	252.4	787.3	40.3	6938.3	19326.1
qbfML	10.5	29.5	409.6	3151.9	10066060.9	44051455.1

Table 2: The comparison of necessary possible worlds, SAT variables and clauses between S5cheetah and *S52SAT*.

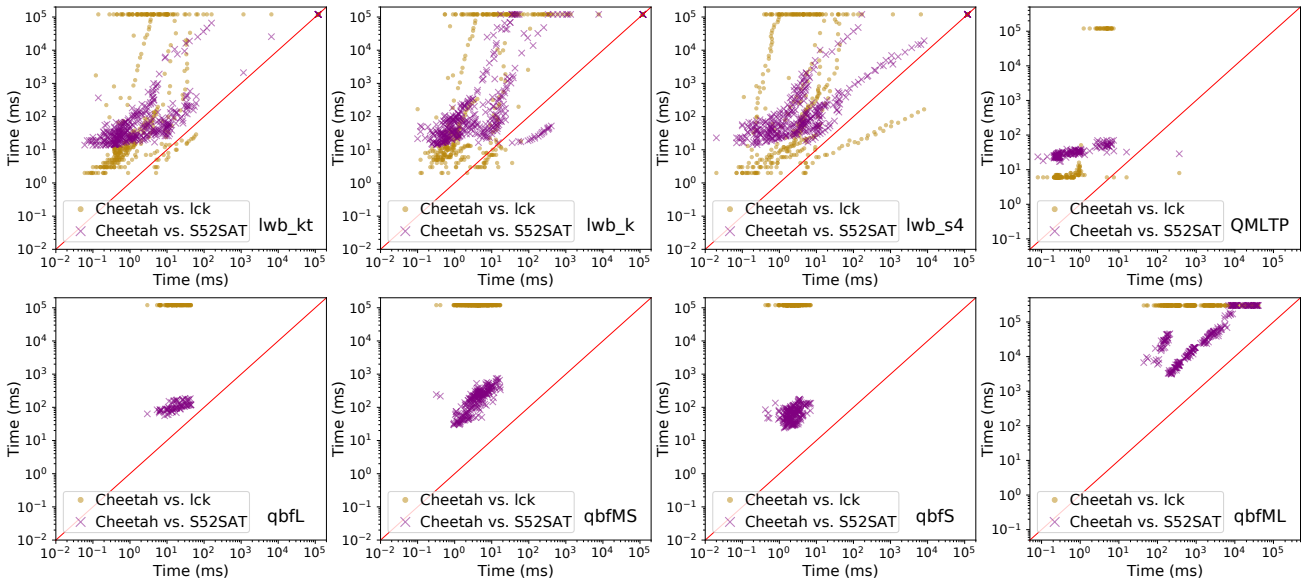

Figure 5: The comparison of run time on LWB, MQBF, TANCS2000 and QMLTP

Instances	S5cheetah		S52SAT		LCK	
	Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
QMLTP	0.3MB	3.7MB	4.8MB	100MB	5.9GB	21.2GB
QS5	124.0MB	1.4GB	8.5GB	50.0GB	-	-
3CNF	67.5MB	1.1GB	440MB	1.6GB	-	-
LWB_k	8.4MB	100.0MB	567.9MB	1.2GB	385MB	6.9GB
LWB_kt	3.0MB	162.0MB	788.5MB	20GB	200MB	1.3GB
LWB_s4	5.0MB	84.0MB	4.0GB	20.0GB	200MB	1.2GB
qbfL	3.6MB	6.9MB	10.1MB	46.8MB	-	-
qbfMS	0.4MB	9.5MB	41.3MB	90.5MB	-	-
qbfS	2.9MB	5MB	28.6MB	41.9MB	-	-
qbfML	50.1MB	100.0MB	10.0GB	49.0GB	-	-

Table 3: The memory usage of three solvers

The number of SAT clauses can reflect memory usage from a certain angle. However, it does not mean actual memory usage. The solver may store some intermediate information while running. Besides, *LCK* is not based on SAT. So, we record the maximum memory usage for each instance family and the results are shown in Table 3.

We did not record the *LCK* memory consumption on $TANCS$, $QS5$ and $3CNF$ since *LCK* is barely able to solve any instance in these benchmarks. The results in Table 3 show that S5cheetah uses much less memory than other solvers. The maximum memory usage of S5cheetah for these benchmarks is 1.4GB. By contrast, *S52SAT* often consumes more than 1GB memory on many benchmarks and even 50GB in some extreme circumstances. Therefore, our solver is much more practical in terms of memory consumption.

8 Conclusion

We propose a novel SAT-based method to solve the S5-SAT problem and implement a fast solver called S5cheetah. First, an input S5 formula is normalized to a kind of first degree normal form, namely, S5-NF. Apart from reducing the modal degree of the original S5 formula, it can make structural and semantic information more obvious. So, a reasoning method for the candidate Kripke model comes into being. We design a strategy called diamond conflict checking (DCC) which can make good use of the structural and semantic information of the S5-NF to eliminate a lot of unnecessary search spaces and automatically discover the small-model potential of a modal logic formula.

The experiments show that our S5-SAT solver is much more efficient than some state-of-the-art S5 solvers and it consumes less memory. This significant improvement inspires us to improve other methods (e.g. Tableau) to tackle S5-SAT and study how to migrate this idea to other modal logics in the future work.

Acknowledgments

This work is supported by the Key Research Program of Frontier Sciences, Chinese Academy of Sciences (CAS), Grant No.QYZDJ-SSW-JSC036, and National Natural Science Foundation of China (NSFC) under grant No.61672504. Feifei Ma is also supported by the Youth Innovation Promotion Association, CAS.

We would like to thank the anonymous reviewers for their comments and suggestions. We thank Yongmei Liu and Qiang Liu for providing some benchmarks.

References

[Abate *et al.*, 2007] Pietro Abate, Rajeev Goré, and Florian Widmann. Cut-free single-pass tableaux for the logic of common knowledge. In *Workshop on Agents and Deduction at TABLEAUX*, volume 2007. Citeseer, 2007.

[Aguilera and Fernández-Duque, 2016] Juan P. Aguilera and David Fernández-Duque. Verification logic: An arithmetical interpretation for negative introspection. In *Proc. of AiML '16*, pages 1–20, 2016.

[Auffray *et al.*, 1990] Yves Auffray, Patrice Enjalbert, and Jean-Jacques Hébrard. Strategies for modal resolution:

Results and problems. *J. Autom. Reasoning*, 6(1):1–38, 1990.

[Balco *et al.*, 2018] Samuel Balco, Sabine Frittella, Giuseppe Greco, Alexander Kurz, and Alessandra Palmigiano. Software tool support for modular reasoning in modal logics of actions. In *Proc. of ITP '18*, pages 48–67, 2018.

[Balsiger *et al.*, 2000] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. A benchmark method for the propositional modal logics k, kt, S4. *J. Autom. Reasoning*, 24(3):297–317, 2000.

[Bienvenu *et al.*, 2010] Meghyn Bienvenu, Hélène Fargier, and Pierre Marquis. Knowledge compilation in the modal logic S5. In *Proc. of AAAI '10*, 2010.

[Caridroit *et al.*, 2017] T. Caridroit, J. Lagniez, D. Le Berre, T. de Lima, and V. Montmirail. A sat-based approach for solving the modal logic s5-satisfiability problem. In *Proc. of AAAI '17*, pages 3864–3870, 2017.

[del Cerro, 1982] Luis Fariñas del Cerro. A simple deduction method for modal logic. *Inf. Process. Lett.*, 14(2):49–51, 1982.

[Enjalbert and del Cerro, 1989] Patrice Enjalbert and Luis Fariñas del Cerro. Modal resolution in clausal form. *Theor. Comput. Sci.*, 65(1):1–33, 1989.

[Fagin *et al.*, 2004] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[Fairtlough and Mendler, 1994] Matt Fairtlough and Michael Mendler. An intuitionistic modal logic with applications to the formal verification of hardware. In *Proc. of CSL '94*, pages 354–368, 1994.

[Fitting, 1999] Melvin Fitting. A simple propositional S5 tableau system. *Ann. Pure Appl. Logic*, 96(1-3):107–115, 1999.

[Gasquet *et al.*, 2005] Olivier Gasquet, Andreas Herzig, Dominique Longin, and Mohamad Sahade. Lotrec: Logical tableaux research engineering companion. In *Proc. of TABLEAUX '05*, pages 318–322, 2005.

[Giunchiglia and Sebastiani, 1996] Fausto Giunchiglia and Roberto Sebastiani. Building decision procedures for modal logics from propositional decision procedure - the case study of modal K. In *Proc. of CADE '96*, pages 583–597, 1996.

[Götzmann *et al.*, 2010] Daniel Götzmann, Mark Kaminski, and Gert Smolka. Spartacus: A tableau prover for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 262:127–139, 2010.

[Kaminski and Tebbi, 2013] Mark Kaminski and Tobias Tebbi. Inkresat: Modal reasoning via incremental reduction to SAT. In *Proc. of CADE '13*, pages 436–442, 2013.

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

[Lagniez *et al.*, 2018] Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, and Valentin Montmirail. A sat-based approach for PSPACE modal logics. In *Proc. of KR '18*, pages 651–652, 2018.

[Lorini and Schwarzentruber, 2010] Emiliano Lorini and François Schwarzentruber. A modal logic of epistemic games. *Games*, 1(4):478–526, 2010.

[Massacci and Donini, 2000] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-2000 non-classical (modal) systems comparison. In *Proc. of TABLEAUX '00*, pages 52–56, 2000.

[Massacci, 1999] Fabio Massacci. Design and results of the tableaux-99 non-classical (modal) systems comparison. In *Proc. of TABLEAUX '99*, pages 14–18, 1999.

[Mints and Minc, 1992] G. Mints and G. E Minc. *Short Introduction to Modal Logic*. Center for the Study of Language, 1992.

[Nalon *et al.*, 2017] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. KSP: A resolution-based prover for multimodal k, abridged report. In *Proc. of IJCAI '17*, pages 4919–4923, 2017.

[Niveau and Zanuttini, 2016] Alexandre Niveau and Bruno Zanuttini. Efficient representations for the modal logic S5. In *Proc. of IJCAI '16*, pages 1223–1229, 2016.

[Ohlbach, 1991] Hans Jürgen Ohlbach. Semantics-based translation methods for modal logics. *J. Log. Comput.*, 1(5):691–746, 1991.

[Patel-Schneider and Sebastiani, 2003] Peter F. Patel-Schneider and Roberto Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *J. Artif. Intell. Res.*, 18:351–389, 2003.

[Robinson and Voronkov, 2001] Alan JA Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 2. Elsevier and MIT Press, 2001.

[Salhi and Sioutis, 2015] Yakoub Salhi and Michael Sioutis. A resolution method for modal logic S5. In *Proc. of GCAI '15*, pages 252–262, 2015.

[Tseitin, 1968] Grigori Tseitin. On the complexity of derivation in propositional calculus. *J. Stud. const. math. and mathematic. logic*, pages 115–125, 1968.

[Wan *et al.*, 2015] Hai Wan, Rui Yang, Liangda Fang, Yongmei Liu, and Huada Xu. A complete epistemic planner without the epistemic closed world assumption. In *Proc. of IJCAI '15*, pages 3257–3263, 2015.

[Welsh and Powell, 1967] Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *Comput. J.*, 10(1):85–86, 1967.