# Tree-Based Heuristics in Modal Theorem Proving

**E0170**

**Abstract.** We use a strong form of the tree model property to boost the performance of resolution-based first-order theorem provers on the so-called relational translations of modal formulas. We provide both the mathematical underpinnings and experimental results concerning our improved translation method.

**Keywords:** automated reasoning, theorem proving, modal reasoning, tree model property.

## 1 Introduction

Modal and modal-like logics such as temporal logic, description logic, and feature logic, have had a long history in artificial intelligence, both as an area of foundational research and as a source for useful representation formalisms and reasoning methods [6, 9].

The recent advent of agent-based technologies has dramatically increased the need for efficient automated reasoning methods for modal logic [6]. Broadly speaking, there are three general strategies for modal theorem proving: (1) develop purpose-built calculi and tools; (2) translate modal problems into automata-theoretic problems, and use automata-theoretic methods to obtain answers; and (3) translate modal problems into first-order problems, and use general first-order tools. The advantage of indirect methods such as (2) and (3) is that they allow us to re-use well-developed and well-supported tools instead of having to develop new ones from scratch.

In this paper we focus on the third option: translation-based theorem proving for modal logic, where modal formulas and reasoning problems are translated into first-order formulas and reasoning problems to be fed to first-order theorem provers. Our starting point is the *standard* or *relational translation* [2, 10], which translates modal formulas by transcribing their truth definitions in first-order terms. First-order theorem provers perform poorly on the outputs of this translation [10]. To overcome this, very sophisticated decision procedures have been developed [5], and alternative translations have been proposed [10]. In this paper, we describe a very intuitive and effective heuristic for modal theorem proving that can be implemented on top of existing strategies and procedures. Briefly, we propose a syntactic encoding of the fact that many modal languages enjoy a very strong form of the so-called tree model property: a modal formula is satisfiable (or more precisely: $\mathbf{K}$-satisfiable) if and only if it is satisfiable at the root of a model based on a tree.

Below, we start by explaining why plain resolution is not a decision procedure for relational translations of modal formulas. To motivate our proposed solution we then explain the tree model property, and recall some basic facts about it. In Section 4 we exploit the tree model property in a new relational translation of modal formulas into first-order logic that encodes the tree model property. Then, in Section 5, we report on our experimental work with the new translation; we have run our tests using the modal QBF benchmark developed within the TANCS [11] competition on theorem proving and satisfiability testing for non-classical logics. Problems were fed to Bliksem [4], a general first-order theorem prover. We conclude with a discussion and plans for future work.

## 2 Background

We start by explaining in more detail why plain resolution is not a decision procedure for relational translations of modal formulas. We begin by recalling the relational translation.

Let $Prop$ be a non-empty set of proposition letters. Formulas of the uni-modal language $\mathcal{ML}$ are built up from proposition letters $p \in Prop$, using $\neg$, $\wedge$, and the modal operator $\Diamond$. Let $Index$ be some index set. Formulas of the multi-modal language $\mathcal{MML}$ are built up from proposition letters $p \in Prop$, using $\neg$, $\wedge$, and modal operators $\langle a \rangle$, for $a \in Index$. The vocabulary of the first-order language $\mathcal{FO}_1$ has unary predicate symbols $P$ corresponding to the proposition letters in $Prop$, and a single binary relation symbol $R$. Instead of a single binary relation symbol $R$, the vocabulary of the first-order language $\mathcal{FO}_2$ has binary relation symbols $R_a$, for every $a \in Index$.

*Models* for $\mathcal{ML}$ are structures of the form $\mathcal{M} = (W, R, V)$, where $W$ is a non-empty domain, $R$ is a binary relation on $W$, and $V$ is a function that takes a proposition letter to the set of states (elements of $W$) where it is true. Truth is defined relative to a state in a model. The important case is $\mathcal{M}, w \models \Diamond\phi$ iff there exists $v$ in $\mathcal{M}$ with $\mathcal{M}, v \models \phi$ and $Rwv$. Models for $\mathcal{ML}$ are structures $(W, \{R_a \mid a \in Index\}, V)$ where each modal operator $\langle a \rangle$ is interpreted using its own binary relation $R_a$. Models for $\mathcal{ML}$ and $\mathcal{MML}$ can also be viewed as models for the corresponding first-order languages $\mathcal{FO}_1$ and $\mathcal{FO}_2$, respectively. To interpret the unary predicate symbols, we look up the values of the corresponding proposition letters in the valuation.

**Definition 2.1 (Relational Translation)** The *relational translation* $ST(\phi)$ of uni-modal formulas $\phi$ into first-order formulas of $\mathcal{FO}_1$, is defined as follows. Let $x$ be an individual variable.

$$
\begin{aligned}
ST_x(p) &= P(x) && (1)\\
ST_x(\neg\phi) &= \neg ST_x(\phi) &&\\
ST_x(\phi \wedge \psi) &= ST_x(\phi) \wedge ST_x(\psi) &&\\
ST_x(\Diamond\phi) &= \exists y\,(Rxy \wedge ST_y(\phi)). && (2)
\end{aligned}
$$

In (1), $P$ is the unary predicate symbol corresponding to the proposition letter $p$; in (2), the variable $y$ should be fresh. Observe how (2) reflects the truth definition for the modal operator $\Diamond$. The translation $ST$ is easily extended to a translation taking multi-modal formulas into $\mathcal{FO}_2$, by using the relation symbol $R_a$ instead of just $R$ in the translation of the modal operator $\langle a \rangle$.

For example, the modal formula $\Box(p \rightarrow \Diamond p)$ translates into the first-order formula $\forall y\,(Rxy \rightarrow (Py \rightarrow \exists z\,(Ryz \wedge Pz)))$.

One can show that a modal formula is satisfiable iff their relational translation is. This effectively embeds the modal languages considered here into first-order languages, and, thus, opens the way to solving modal problems by first-order means. The resulting first-order fragments can be described as follows.

**Definition 2.2 (Fragments)** Let $x$ be an individual variable. The *modal fragment* MF of $\mathcal{FO}_1$ is built up from unary atoms $Px$, using negation, conjunction, and guarded quantifications of the form $\exists y\,(Rxy \wedge \alpha[x \mapsto y])$, and $\forall y\,(Rxy \rightarrow \alpha[x \mapsto y])$, where $y$ is fresh, and $\alpha[x \mapsto y]$ is the result of replacing all free occurrences of $x$ in $\alpha$ by $y$, and $\alpha(x) \in$ MF only has $x$ free.

The *guarded fragment* GF was proposed as a generalization of the modal fragment [1], where the limitation to unary and binary predicate symbols has been lifted. Every atomic formula $R\overline{xy}$ or $x_i = x_j$ belongs to GF, and GF is closed under the booleans. If $\overline{x}, \overline{y}$ are tuples of variables, $R\overline{xy}$ is an atomic formula and $\phi(\overline{x}, \overline{y})$ is a formula in GF such that all free variables of $\phi$ occur in $R$, then $\exists \overline{y}\,(R\overline{xy} \wedge \phi(\overline{x}, \overline{y}))$ and $\forall \overline{y}\,(R\overline{xy} \rightarrow \phi(\overline{x}, \overline{y}))$ belong to GF.

Observe that the relational translation maps modal formulas into MF, and that the guarded fragment extends MF.

**Example 2.3** Consider the formula $\Box(p \rightarrow \Diamond p)$ again; it is clearly satisfiable. Proving this in first-order logic amounts to showing that the following set of clauses is satisfiable.

1. $\{\neg R(c, y),\ \neg P(y),\ R(y, f(y))\}$
2. $\{\neg R(c, z),\ \neg P(z),\ P(f(z))\}$

The clauses have two resolvents ($f^n$ is $f$ applied $n$ times):

3. $\{\neg R(c, c),\ \neg P(c),\ \neg P(f(c)),\ P(f(f(c)))\}$
4. $\{\neg R(c, f(z)),\ R(f(z), f^2(z)),\ \neg R(c, z),\ \neg P(z)\}$.

Clauses 2 and 4 resolve to produce

5. $\{\neg R(c,f^2(z)), R(f^2(z),f^3(z)), \neg R(c,f(z)), \neg R(c,z), \neg P(z)\}$.

Clauses 2 and 5 resolve again to produce an analogue of 5 with even higher term-complexity, etc. None of the clauses is redundant and can be deleted; in the limit our input set has infinitely many resolvents. This shows that standard resolution does not necessarily terminate for relational translations of satisfiable modal formulas.

What went wrong in Example 2.3? First, to obtain the resolvent in line 3, a positive and negative binary literal were resolved; note that these literals (or rather: the modal operators from which they derive) live at different modal depths in the original modal formula $\Box(p \rightarrow \Diamond p)$. This resolution step is useless: the negative $R$-literal derives from the $\Box$-operator which occurs at modal depth 0, and the positive $R$-literal comes from the $\Diamond$-operator which occurs at modal depth 1. Unless we explicitly stipulate so (by means of axioms), different modal depths are completely independent and cannot resolve. Second, a similar comment can be made about the resolvent obtained in line 4, where a positive and negative unary literal corresponding to the two occurrences of the proposition letter $p$ were resolved upon.

We will boost the performance of resolution procedures on the relational translation of modal formulas by making literals living at different modal depths syntactically different. The mathematical justification for these ideas is provided by a strong form of the tree model property, as we will explain in the following section.

## 3 The Tree Model Property

To increase the performance of general first-order theorem provers on 'modal input,' we will feed the provers with information about the modal character of the input. More precisely, we will aim to encode by syntactic means the fact that basic modal logic enjoys a very strong form of the *tree model property*. In recent years, the latter has been identified as one of the semantic key features explaining the good logical and computational behavior of many modal logics; see [7, 12] for two very accessible presentations.

First, by a *tree* $\mathcal{T}$ we mean a relational structure $(T, S)$ where $T$, the set of nodes, contains a unique $r \in T$ (called the *root*) such that $\forall t \in T\,(S^* rt)$; every element of $T$ distinct from $r$ has a unique $S$-predecessor; and $S^+$ is acyclic; that is, $\forall t\,(\neg S^+ tt)$. (Here, $S^+$ and $S^*$ denote the transitive and reflexive, transitive closure of $S$, respectively.)

A *tree model* (for the uni-modal language $\mathcal{ML}$) is a model $\mathcal{M} = (W, R, V)$, where $(W, R)$ is a tree. A *tree-like model* for the multi-modal language $\mathcal{MML}$ is a model $(W, \{R_a \mid a \in Index\}, V)$ such that $(W, \bigcup_a R_a)$ is a tree. A logic $\mathbf{L}$ has the *tree model property* if every $\mathbf{L}$-satisfiable formula is satisfiable at the root of a tree or tree-like model for $\mathbf{L}$. Observe that the tree model property is incomparable to the finite model property; there are modal logics where the former fails but the latter holds, and vice versa.

We refer the reader to any introduction to modal logic for definitions of the basic uni-modal logic $\mathbf{K}$ and the basic multi-modal logic $\mathbf{K}_{(m)}$; see [3], for instance.

**Proposition 3.1** *1. The basic uni-modal logic* $\mathbf{K}$ *has the tree model property.*
*2. The basic multi-modal logic* $\mathbf{K}_{(m)}$ *has the tree model property.*

Many modal logics, including $\mathbf{K}$ and $\mathbf{K}_{(m)}$, enjoy stronger versions of the tree model property, where the degree of the tree model can be bounded by the size of the formula [3]. But $\mathbf{K}$ and $\mathbf{K}_{(m)}$ enjoy an even stronger version of the tree model property. The key notion here is that of *layering*, both w.r.t. tree models and w.r.t. formulas. Tree (or tree-like) models come with a layering induced by the *depth* of the nodes. Likewise, the parse tree of a modal formula induces a natural formula layering, where new layers begin immediately below nodes labeled by modal operators. For instance, in $\Box(p \rightarrow \Diamond p)$, the $\Box$ occurs in layer 0, while the $\Diamond$ occurs in layer 1, with its argument in layer 2. Next, the *modal depth*, mdepth($\phi$), of a uni-modal or multi-modal formula $\phi$ is defined as follows. Proposition letters $p$ have mdepth($p$) $= 0$; mdepth($\neg\psi$) $=$ mdepth($\psi$); mdepth($\psi \wedge \chi$) $= \max($mdepth($\psi$), mdepth($\chi$)$)$, while mdepth($\Diamond\psi$) $=$ mdepth($\langle a\rangle\psi$) $= 1 +$ mdepth($\psi$).

**Proposition 3.2** *Let $\phi$ be a modal formula, and $\mathcal{M}$ be a tree (or tree-like) model with root $w$ such that $\mathcal{M}, w \models \phi$.*

*Let $\psi$ be a subformula of $\phi$ which occurs in formula layer $l$ and which has modal depth $k$. To determine the truth value of $\psi$ we only need to consider nodes at tree depth $i$, where $l \leq i \leq k + l$.*

In words: there is a direct correlation between formula layers and layers in a tree (or tree-like) model; as a consequence, literals occurring at different formula layers cannot resolve and need not be combined.

## 4 Boosting the Relational Translation

In this section we exploit the tree-based intuitions developed in the previous section, and propose a new relational translation of modal

formulas into first-order formulas, one that tries to encode the fact that modal formulas enjoy the tree model property.

We will proceed in two steps: we will first translate into an intermediate modal language, and from there into first-order logic; the latter step will use the relational translation of Definition 2.1. We start by defining the translation process for uni-modal formulas; after that we give the translation for multi-modal formulas.

**From Uni-Modal to First-Order.** The key idea behind our translation is to label unary and binary relations according to the *number* of modal operators nested within a modal formula. For instance, the formula $p$ is translated into $P_0(x)$, while the formula $\Diamond p$ becomes $\exists y\,(R_1(x,y) \wedge P_1(y))$. The index 1 of the relation symbols $R_1$ and $P_1$ measures the modal depth of the modal formula.

To motivate the translation of uni-modal $\mathcal{ML}$ formulas into an intermediate multi-modal language, consider the following examples, where we use new operators and new proposition letters each time we change modal depth:

$$\begin{array}{rcl} \Diamond\Diamond p & \mapsto & \Diamond_1\Diamond_2 p_2 \\ \Box(p \rightarrow \Diamond p) & \mapsto & \Box_1(p_1 \rightarrow \Diamond_2 p_2). \end{array}$$

If we then apply the relational translation (Definition 2.1) to the intermediate multi-modal representations, we obtain $\exists y\,(R_1 xy \wedge \exists x(R_2 yx \wedge P_2 x))$ and $\forall y\,(P_1 y \rightarrow \exists x\,(R_2 yx \wedge P_2 x))$, respectively. Observe that the problematic derivation from the normal relational translation of $\Box(p \rightarrow \Diamond p)$ in Example 2.3 is no longer possible with the new first-order translation.

To make things precise, we need an intermediate multi-modal language $\mathcal{IML}_1$, whose collection of modal operators is $\{\Diamond_i \mid i \geq 0\}$.

**Definition 4.1** Let $\phi$ be a uni-modal formula. Let $n$ be a natural number. The translation $Tr(\phi, n)$ of $\phi$ into the intermediate modal language $\mathcal{IML}_1$ is defined as follows:

$$\begin{array}{rcl} Tr(p, n) & := & p_n \\ Tr(\neg\psi, n) & := & \neg\, Tr(\psi, n) \\ Tr(\psi \wedge \chi, n) & := & Tr(\psi, n) \wedge Tr(\chi, n) \\ Tr(\Diamond\psi, n) & := & \Diamond_{n+1}\, Tr(\psi, n+1). \end{array}$$

Our next aim is to show that the intermediate translation $Tr$ preserves satisfiability.

**Lemma 4.2** *Let $\phi$ be a uni-modal formula. If $\phi$ is satisfiable, then so is its intermediate multi-modal translation $Tr(\phi, 0)$.*

*Proof.* By Proposition 3.1 we may assume that $\phi$ is satisfiable at the root $w$ of a tree model $\mathcal{M} = (W, R, V)$. Since $\mathcal{M}$ is a tree model, for every state $v \in W$ there exists a unique path of $R$-steps from the root $w$ to $v$; let $d(w, v)$ denote the length of this path.

We define a model $\mathcal{N} = (W, \{R_{n+1} \mid n \geq 0\}, V')$ for the intermediate multi-modal language $\mathcal{IML}_1$ by taking its universe to be $W$, the universe of $\mathcal{M}$. Its relations are defined by stipulating that $R_{n+1}(u, v)$ holds iff $d(w, u) = n$ and $R(u, v)$ both hold. We complete the definition of $\mathcal{N}$ by defining the valuation $V'$: for every proposition letter $p$ and every state $v \in W$ such that $d(w, v) = n$, we put $v \in V'(\,Tr(p, n))$ iff $v \in V(p)$.

We leave it to the reader to show that for every uni-modal formula $\phi$, every state $v$ and every $n$ such that $d(w, v) = n$, we have $\mathcal{M}, v \models \phi$ iff $\mathcal{N}, v \models Tr(\phi, n)$. From this the lemma follows. $\dashv$

**Lemma 4.3** *Let $\phi$ be a uni-modal formula. If its intermediate multi-modal translation $Tr(\phi, 0)$ is satisfiable, then so is $\phi$.*

*Proof.* Let $Tr(\phi, 0)$ be satisfied at some state $w$ in some model $\mathcal{M}$ for the intermediate multi-modal language $\mathcal{IML}_1$. As before we may assume that $\mathcal{M}$ is a tree-like model with root $w$. We define a uni-modal model $\mathcal{N}$ which differs from $\mathcal{M}$ in that it has only one relation ($R$) and in its valuation. The relation $R$ consists of all pairs $(u, v)$ such that $(u, v) \in R_{n+1}$ and $d(w, u) = n$, where $d(w, u)$ is the length of the path $w$ to $u$ (in $\mathcal{M}$). The valuation $V'$ of our model $\mathcal{N}$ is defined as follows: for every proposition letter $p$, for every $v$ such that $d(w, v) = n$, we put $v \in V(p)$ iff $v \in V(\,Tr(p, n))$, where $V$ is $\mathcal{M}$'s valuation. One can then show that if $d(w, v) = n$, then $\mathcal{M}, v \models Tr(\phi, n)$ iff $\mathcal{N}, v \models \phi$. This implies the lemma. $\dashv$

**Theorem 4.4** *Let $\phi$ be a uni-modal formula. Then $\phi$ is satisfiable iff its intermediate multi-modal translation $Tr(\phi, 0)$ is.*

**Definition 4.5** The *layered relational translation* is the composition of $Tr$ and $ST$.

**Theorem 4.6** *Let $\phi$ be a uni-modal formula. Then $\phi$ is satisfiable iff its layered relational translation $ST(\,Tr(\phi, 0))$ is.*

**From Multi-Modal to First-Order.** We now extend the layered translation to the multi-modal language $\mathcal{MML}$; again, we go through an intermediate multi-modal language. The basic ideas are the same as in the uni-modal case, but the presence of multiple modal operators in the source language calls for changes. The set of operators of the intermediate language $\mathcal{IML}_2$ is $\{\Diamond_s \mid s \in Index^*\}$.

**Definition 4.7** Let $\phi$ be a multi-modal formula in $\mathcal{MML}$. Let $s \in Index^*$. The translation $Tr(\phi, s)$ of $\phi$ into the intermediate multi-modal language $\mathcal{IML}_2$ is defined by:

$$\begin{array}{rcl} Tr(p, s) & := & p_s \\ Tr(\neg\psi, s) & := & \neg\, Tr(\psi, s) \\ Tr(\psi \wedge \chi, s) & := & Tr(\psi, s) \wedge Tr(\chi, s) \\ Tr(\langle a\rangle\psi, s) & := & \Diamond_{s*\langle a\rangle}\, Tr(\psi, s * \langle a\rangle). \end{array}$$

Using the tree model property we establish the following theorem:

**Theorem 4.8** *Let $\phi$ be a multi-modal formula. Then $\phi$ is satisfiable iff its intermediate multi-modal translation $Tr(\phi, \epsilon)$ is.*

**Theorem 4.9** *Let $\phi$ be a multi-modal formula. Then $\phi$ is satisfiable iff its layered relational translation $ST(\,Tr(\phi, \epsilon))$ is.*

**Comments.** With the layered relational translation we have obtained a new way of turning modal problems into first-order problems. The layered translation is conservative in the sense that it can work *on top of* existing strategies. In particular, the layered translation maps modal formulas into the guarded fragment, thus we can use existing decision procedures for GF [5].

**Theorem 4.10** *Let $\mathcal{R}_S(\phi)$ and $\mathcal{R}_L(\phi)$ denote the sets of clauses derivable from $ST(\phi)$ and $ST(\,Tr(\phi, 0))$ or $ST(\,Tr(\phi, \epsilon))$, respectively, by means of binary resolution and factoring. Then $|\mathcal{R}_L(\phi)| \leq |\mathcal{R}_S(\phi)|$.*

So, the layered translation will perform at least as well as the standard translation. In Section 5 below we report on our experiments which actually show a dramatic improvement of the layered translation over the standard one.

## 5 Experimental Results

We will now put our new relational translation to the test. Before going into the test results, we comment on the problem set and theorem prover used in our experiments.

**The Problem Set.** To evaluate our tree-based heuristics, we have run a series of tests using the modal QBF benchmark. This benchmark is the basic yardstick for the TANCS (Tableaux Non-Classical Systems Comparisons) competition on theorem proving and satisfiability testing for non-classical logics [11]. It is a random problem generator that has been designed for evaluating solvers of (un-) satisfiability problems for the modal logic **K**.

The formulas of this benchmark are generated using quantified boolean formulas. For the generation, a quantified boolean formula with $C$ clauses is generated with a quantifier alternation depth $D$, and for each alternation at most $V$ variables are used. The resulting formula is then translated into modal logic using the encoding originally proposed by Halpern [8]. The output of the QBF generator is a file named p-qbf-cnf-K4-C$n$-V$m$-D$l$, where $n$, $m$, $l$ stand for the number of clauses, variables and depth respectively.

**The Theorem Prover.** Tests were performed on a Sun ULTRA II (300MHz) with 1Gb RAM, under Solaris 5.2.5, with Bliksem version 1.10a. Bliksem is a general purpose theorem prover for first-order logic that has been developed by Hans de Nivelle [4]. It uses resolution plus non-liftable orderings with weak subsumption, which are able do decide many fragments of first-order logic. In particular, Bliksem is able to recognize and decide formulas inside the guarded fragment. The fact that Bliksem always terminates when deciding a guarded formula, made it especially appropriate for our tests. But our translation should improve the behavior of *any* resolution-based prover compared to the standard translation.

**Results.** To explore the behavior of our heuristics in a large portion of the landscape of the **K**-satisfiability problem, we randomly generated sets of 10 problems by means of QBF for different sets of parameters. Table 1 compares the average time and number of clauses for two methods: *layered* (our improved translation) and *standard* (the relational method). "C/D/V" in the first column denotes the number of clauses, depth and number of variables used in the generation. Columns labeled by "M" show the magnitude of the difference between the preceding two columns, i.e., $\mathrm{round}(-1 * log(N/N'))$.

As can easily be seen from Table 1, our improved translation method outperformed the standard translation in every case, both in computing time (CPU time) and number of clauses generated; this is not only an average behavior but it was observed in each instance. For some configurations (5/2/4,10/2/4,10/3/2,15/3/2) the drop in computing time is as much as four orders of magnitude, with the largest difference occurring with 15/3/2: from an average of 8 hours and 40 minutes down to around 7 seconds! The average number of clauses generated was nearly always smaller by one order of magnitude.

In Figure 1 we display a selection of our experimental results to highlight some interesting phenomena. In each case we plot the result for 10 randomly generated formulas, showing the number of clauses needed to obtain a refutation in the top graph, and the CPU time needed to find the proof in the bottom graph. In Example (a), we see that the standard translation can be very sensitive to certain hard problems like instances 1 and 2. Example (b) shows an important difference in computing time between the standard and layered methods. Interestingly, the curves follow each other, even at many orders of magnitude of difference. This shows that our heuristics does not change the nature of the problem; it simply makes it much easier for the resolution prover. The graphs in Example (c) show a similar behavior with the layered method responding in a much more controlled way to hard problems (as we can see in instances 3 and 7).

| C/D/V | Average Time | | M | Average Clauses | | M |
|---|---|---|---|---|---|---|
| | Layered | Standard | | Layered | Standard | |
| 5/2/1 | 0.238 | 0.277 | 0 | 192 | 676 | 1 |
| 10/2/1 | 0.381 | 0.423 | 0 | 336 | 787 | 0 |
| 15/2/1 | 0.545 | 0.582 | 0 | 401 | 852 | 0 |
| 5/2/2 | 0.799 | 12.010 | 1 | 970 | 6634 | 1 |
| 10/2/2 | 1.092 | 33.942 | 1 | 1243 | 10073 | 1 |
| 15/2/2 | 1.360 | 38.043 | 1 | 1470 | 10689 | 1 |
| 5/2/3 | 1.496 | 1.873 | 0 | 586 | 1822 | 0 |
| 10/2/3 | 1.922 | 6.119 | 1 | 763 | 5743 | 1 |
| 15/2/3 | 2.418 | 37.281 | 1 | 949 | 11558 | 1 |
| 5/2/4 | 5.140 | 27294.000 | 4 | 4117 | 94581 | 1 |
| 10/2/4 | 7.319 | 23625.000 | 4 | 5070 | 90663 | 1 |
| 15/2/4 | 10.087 | 9536.800 | 3 | 5948 | 72433 | 1 |
| 5/2/5 | 6.887 | 1132.800 | 2 | 1815 | 11346 | 1 |
| 10/2/5 | 9.186 | 1615.000 | 2 | 2198 | 18224 | 1 |
| 15/2/5 | 10.716 | 190.870 | 1 | 1992 | 21093 | 1 |
| 5/3/1 | 0.608 | 0.757 | 0 | 341 | 1220 | 1 |
| 10/3/1 | 0.878 | 2.318 | 0 | 503 | 3853 | 1 |
| 15/3/1 | 1.138 | 3.225 | 0 | 689 | 4518 | 1 |
| 5/3/2 | 3.973 | 8044.600 | 3 | 2936 | 67882 | 1 |
| 10/3/2 | 5.781 | 27107.000 | 4 | 3823 | 109900 | 1 |
| 15/3/2 | 7.271 | 31258.000 | 4 | 4457 | 103250 | 1 |
| 5/3/3 | 6.951 | 2237.400 | 3 | 2245 | 19483 | 1 |
| 10/3/3 | 9.028 | 31.930 | 1 | 1741 | 9548 | 1 |
| 15/3/3 | 10.586 | 154.810 | 1 | 1989 | 20623 | 1 |

**Table 1.** Comparison.

The CPU time curves in Examples (a) and (c) are closer to each other than in Example (b). This was a constant behavior we observed: when the D (depth) parameter was set to an odd value the curves get closer with repeated instances of very hard problems for the standard translation; when the depth was set to an odd value the problems were uniformly much harder for the standard translation. Remember that the D parameter specifies the number of alternations of quantifier blocks in the original QBF formulas. Hence, an odd value stands for a universal formula, while even values produce existential formulas. Clearly, the standard translation is sensible to this, while our layered translation has overcome this problem.

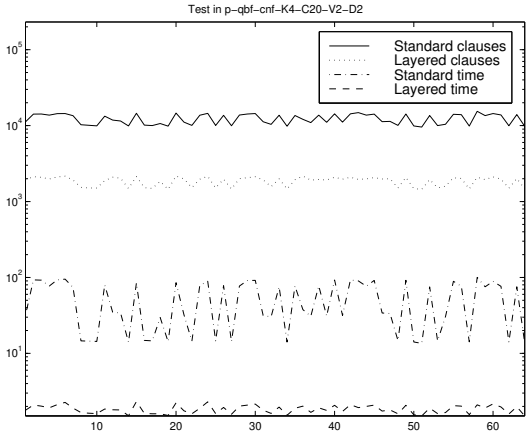
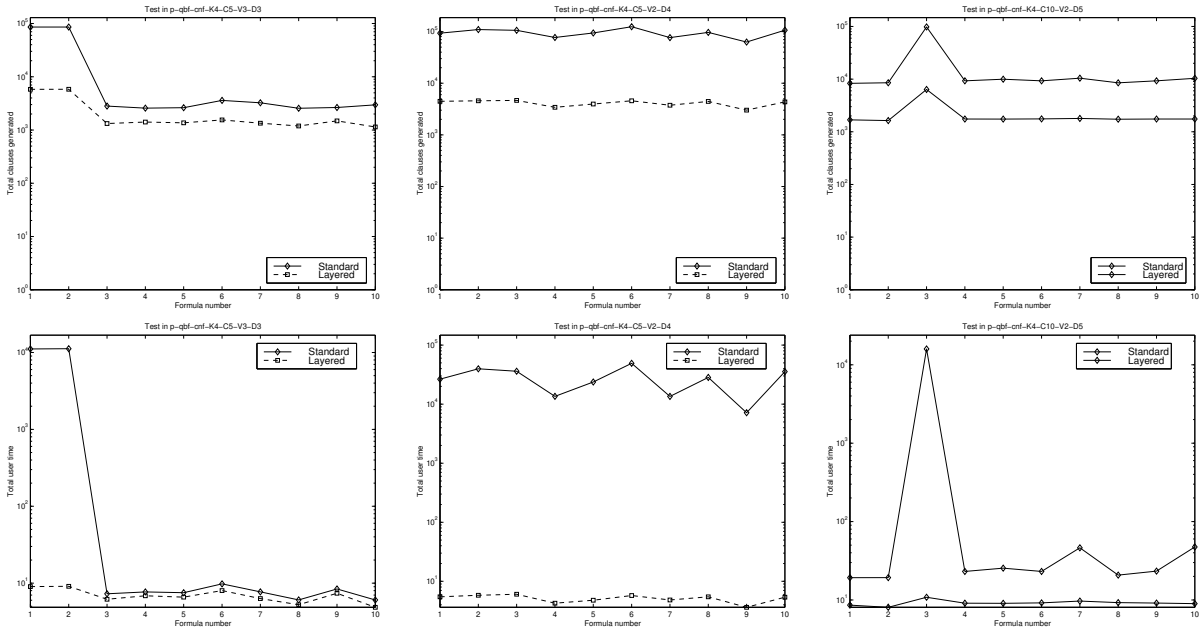



**Figure 2.**




Example (a) Example (b) Example (c)

**Figure 1.** A selection from the experimental tests

To explore a variety of parameters we initially ran only 10 tests for each configuration. Once we were confident that the layered method uniformly displayed good behavior, we ran the standardized tests provided by TANCS (64 instances randomly generated with the 20-clauses/2-variables/2-depth parameters); see Figure 2.

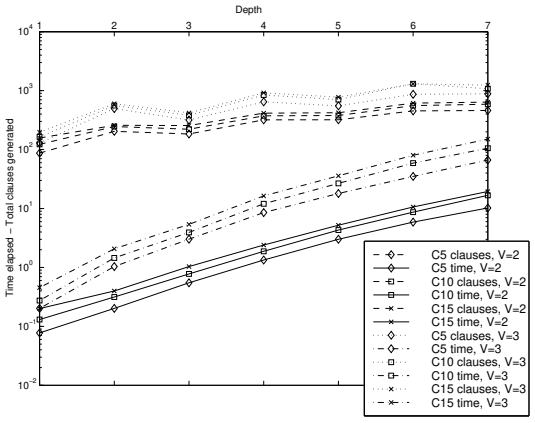



**Figure 3.**

Finally, to obtain the results in Figure 3 we used the random generator to generate 64 problems for 2 and 3 variables with depths ranging from 1 to 7. The figure shows the average values we obtained. We ran the same tests on Bliksem with the standard translation instead of the layered translation, but even for moderate depth (2 or 3) the computing time and number of clauses exceeded the available resources.

## 6 Conclusion

In this paper we have described a new relational translation of modal formulas into first-order formulas. The key idea underlying the improvement is to encode a very strong tree model property in the translation. Using our tree-based heuristics on top of Bliksem's strategy for the guarded fragment, we have consistently outperformed Bliksem with that strategy, both in terms of the number of clauses generated and in terms of CPU time used.

Our ongoing and future work is aimed at extending the test results to other general first-order provers such as SPASS, and at encoding weaker forms of the tree model property to boost the performance of resolution provers on input from different modal logics, such as $\mathbf{K4}$, $\mathbf{S4}$, and temporal logic.

## REFERENCES


[1] H. Andréka, J. van Benthem, and I. Németi, 'Modal languages and bounded fragments of predicate logic', *J. Phil. Logic*, **27**, 217–274, (1998).

[2] J. van Benthem, *Modal Logic and Classical Logic*, Bibliopolis, 1983.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, 1999. Manuscript available from `http://www.illc.uva.nl/~mdr/Publications/modal-logic.html`.

[4] Bliksem Version 1.10B. URL: `http://www.mpi-sb.mpg.de/~bliksem/`. Accessed Jan. 16, 2000.

[5] H. de Nivelle and M. de Rijke, 'Deciding the guarded fragments by resolution', *J. Symb. Comp.*, (to appear).

[6] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi, *Reasoning About Knowledge*, The MIT Press, 1995.

[7] E. Grädel, 'Why are modal logics so robustly decidable?', *Bulletin EATCS*, **68**, 90–103, (1999).

[8] J. Halpern, 'The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic', *Artificial Intelligence*, **75**, 361–372, (1995).

[9] U. Hustadt and R. A. Schmidt, 'On evaluating decision procedures for modal logics', in *Proc. IJCAI'97*, ed., M. Pollack, pp. 202–207, (1997).

[10] H.J. Ohlbach, A. Nonnengart, M. de Rijke, and D.M. Gabbay, 'Encoding two-valued non-classical logics in classical logic', in *Handb. Automated Reasoning*, eds., J. Robinson and A. Voronkov, Elsevier, (1999).

[11] TANCS: Tableaux Non Classical Systems Comparison. URL: `http://www.dis.uniroma1.it/~tancs/`. Accessed Jan. 16, 2000.

[12] M.Y. Vardi, 'Why is modal logic so robustly decidable?', in *DIMACS Ser. Disc. Math. Theoret. Comp. Sci. 31*, 149–184, AMS, (1997).