Tree-Based Heuristics in Modal Theorem Proving

Carlos Areces[1] and **Rosella Gennari**[2] and **Juan Heguiabehere**[1] and **Maarten de Rijke**[1]

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 [4, 8].

The recent advent of agent-based technologies has dramatically increased the need for efficient automated reasoning methods for modal logic [4]. 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* [1, 9], 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 [9]. To overcome this, very sophisticated decision procedures have been developed [3], and alternative translations have been proposed [9]. 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, focusing mainly on our tests with the modal QBF benchmark developed within the TANCS [11] competition on theorem proving and satisfiability testing for non-classical logics. Problems were fed to SPASS [10], a general first-order theorem prover. We conclude with a discussion and plans for future work.

[1] ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands. E-mail: {carlos, juanh, mdr}@wins.uva.nl

[2] Dept. of Philosophy, University of Amsterdam, Nieuwe Doelenstraat 15, 1012 CP Amsterdam, The Netherlands. E-mail: gennari@hum.uva.nl

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 $\mathcal{MML}$ 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 ϕ 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 is 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 $\Box(p \to \Diamond p)$ translates into the first-order formula $\forall y\,(Rxy \to (Py \to \exists z\,(Ryz \wedge Pz)))$; here $\Box$ is short for $\neg\Diamond\neg$.

One can show that a modal formula is satisfiable iff its 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 (Modal Fragment) 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 $\forall y\,(Rxy \to \alpha[x \mapsto y])$, where y is fresh, and $\alpha[x \mapsto y]$ is the result of replacing all free occurrences of x in α by y, and $\alpha(x) \in$ MF only has x free. Observe that the relational translation maps modal formulas into MF.

The modal fragment of $\mathcal{FO}_2$ is defined analogously.

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

1. $\{\neg Rcy,\ \neg Py,\ Ryf(y)\}$
2. $\{\neg Rcz,\ \neg Pz,\ Pf(z)\}$

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

3. $\{\neg Rcc,\ \neg Pc,\ \neg Pf(c),\ Pf(f(c))\}$
4. $\{\neg Rcf(z),\ Rf(z)f^2(z),\ \neg Rcz,\ \neg Pz\}$.

Clauses 2 and 4 resolve to produce

5. $\{\neg Rcf^2(z),\ Rf^2(z)f^3(z),\ \neg Rcf(z),\ \neg Rcz,\ \neg Pz\}$.

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 $\Box(p \to \Diamond p)$. 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 [5, 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 $(W, \{R_a \mid a \in \mathit{Index}\}, V)$ 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 [2], 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 [2]. 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 $\Box(p \to \Diamond p)$, the $\Box$ occurs in layer 0, while the $\Diamond$ occurs in layer 1, with its argument in layer 2. Next, the *modal depth*, mdepth(ϕ), of a uni-modal or multi-modal formula ϕ is defined as follows. Proposition letters p have mdepth(p) $= 0$; mdepth($\neg\psi$) $=$ mdepth(ψ); mdepth($\psi \wedge \chi$) $= \max$(mdepth(ψ), mdepth(χ)), while mdepth($\Diamond\psi$) $=$ mdepth($\langle a \rangle\psi$) $= 1 +$ mdepth(ψ).

Proposition 3.2 *Let ϕ 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 ψ be a subformula of ϕ which occurs in formula layer l and which has modal depth k. To determine the truth value of ψ 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_0x, while the formula $\Diamond p$ becomes $\exists y\,(R_1xy \wedge P_1y)$. 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:

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

If we then apply the relational translation (Definition 2.1) to the intermediate multi-modal representations, we obtain $\exists y\,(R_1xy \wedge \exists z\,(R_2yz \wedge P_2z))$ and $\forall y\,(R_1xy \to (P_1y \to \exists z\,(R_2yz \wedge P_2z)))$, respectively. Observe that the problematic derivation from the standard relational translation of $\Box(p \to \Diamond p)$ 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 ϕ be a uni-modal formula. Let n be a natural number. The translation $Tr(\phi, n)$ of ϕ 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 ϕ be a uni-modal formula. If ϕ is satisfiable, then so is its intermediate multi-modal translation $Tr(\phi, 0)$.*

Proof. By Proposition 3.1 we may assume that ϕ 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 ϕ, 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 ϕ be a uni-modal formula. If its intermediate multi-modal translation $Tr(\phi, 0)$ is satisfiable, then so is ϕ.*

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

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

Theorem 4.5 *Let ϕ be a uni-modal formula. Then ϕ 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 $\{\Diamond_s \mid s \in \mathit{Index}^*\}$.

Definition 4.6 Let ϕ be a multi-modal formula in $\mathcal{MML}$. Let $s \in \mathit{Index}^*$. The translation $Tr(\phi, s)$ of ϕ into $\mathcal{IML}_2$ is given by:

$$\begin{array}{rcl} Tr(p, s) & := & p_s \\ Tr(\neg\psi, s) & := & \neg\, Tr(\psi, s) \\ Tr(\psi \wedge \chi, s) & := & Tr(\psi, s) \wedge Tr(\chi, s) \\ Tr(\langle a\rangle\psi, s) & := & \Diamond_{s*\langle a\rangle}\, Tr(\psi, s * \langle a\rangle). \end{array} \tag{3}$$

In (3) we encode both the modal depth of a subformula as well as the particular modal operator in whose scope it occurs.

Using the tree model property we obtain the following result:

Theorem 4.7 *Let ϕ be a multi-modal formula. Then ϕ is satisfiable iff its intermediate multi-modal translation $Tr(\phi, \epsilon)$ is.*

Theorem 4.8 *Let ϕ be a multi-modal formula. Then ϕ 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 modal fragment, thus we can use any existing decision procedure for MF [3].

Theorem 4.9 *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. Below we report on our experiments which show a dramatic improvement of the layered over the standard translation.

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 scripts used in conducting our experiments are available at `http://www.illc.uva.nl/~mdr/ACLG/Software`.

The Problem Sets. To evaluate our tree-based heuristics, we have run a series of tests on a number of problem sets. Our main focus was on 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 $\mathbf{K}$.

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 an encoding originally proposed by Halpern [6]. The output of the QBF generator is a file named p-qbf-cnf-K4-Cn-Vm-Dl, 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 SPASS version 1.0.3. SPASS is an automated theorem prover for full sorted first-order logic with equality that extends superposition by sorts and a splitting rule for case analysis; it has been in development at the Max-Planck-Institut für Informatik for a number of years [10].

SPASS was invoked with the auto mode switched on; no sort constraints were built, and both optimized and strong Skolemization were disabled.

C/V/D	Average Time		M	Average Clauses		M
	Layered	Standard		Layered	Standard	
5/2/1	0.53469	9.6222	1	726	5695	1
10/2/1	0.41734	3.9909	1	546	2367	1
15/2/1	0.10859	0.13172	0	10	10	0
5/2/2	0.66141	450.44	3	437	27029	2
10/2/2	0.78297	370.09	3	500	22306	2
15/2/2	0.75656	147.38	2	473	11368	1
5/2/3	36.048	N/A	N/A	10714	N/A	N/A
10/2/3	58.996	N/A	N/A	15395	N/A	N/A
15/2/3	94.192	2094.4	1	20786	45798	0
5/2/4	20.362	N/A	N/A	3121	N/A	N/A
10/2/4	33.084	N/A	N/A	4971	N/A	N/A
15/2/4	35.068	N/A	N/A	5358	N/A	N/A
5/2/5	1136.1	N/A	N/A	48546	N/A	N/A
10/2/5	2896	N/A	N/A	91767	N/A	N/A
15/2/5	3758.2	N/A	N/A	106870	N/A	N/A
5/3/1	7.1862	2047.9	2	4372	105960	1
10/3/1	9.752	2324.2	2	5390	108110	1
15/3/1	14.066	1506.8	2	6687	72605	1
5/3/2	7.0931	N/A	N/A	1804	N/A	N/A
10/3/2	8.3192	N/A	N/A	2221	N/A	N/A
15/3/2	9.3902	N/A	N/A	2687	N/A	N/A
5/3/3	1445.2	N/A	N/A	52153	N/A	N/A
10/3/3	4045.1	N/A	N/A	107800	N/A	N/A
15/3/3	4865.4	N/A	N/A	119150	N/A	N/A

Table 1. Comparison.

Results. To explore the behavior of our heuristics in a large portion of the landscape of the $\mathbf{K}$-satisfiability problem, we randomly generated sets of 10 problems by means of QBF for different sets of parameters. Table 1 compares the average time in CPU seconds and number of clauses for two methods: *layered* (our improved translation) and *standard* (the relational method). "C/V/D" in the first column denotes the number of clauses, the number of variables, and the depth used in the generation. Columns labeled by "M" show the magnitude of the difference between the preceding two columns, i.e., $\mathrm{round}(-1 * log(N/N'))$. We used a time out of 3 hours on a shared machine; N/A indicates that a value is not available due to a time out.

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 the drop in computing time is as much as three orders of magnitude. The average number of clauses generated was nearly always smaller by one order of magnitude.

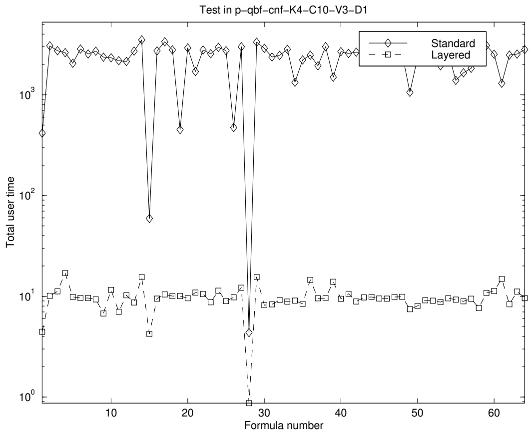
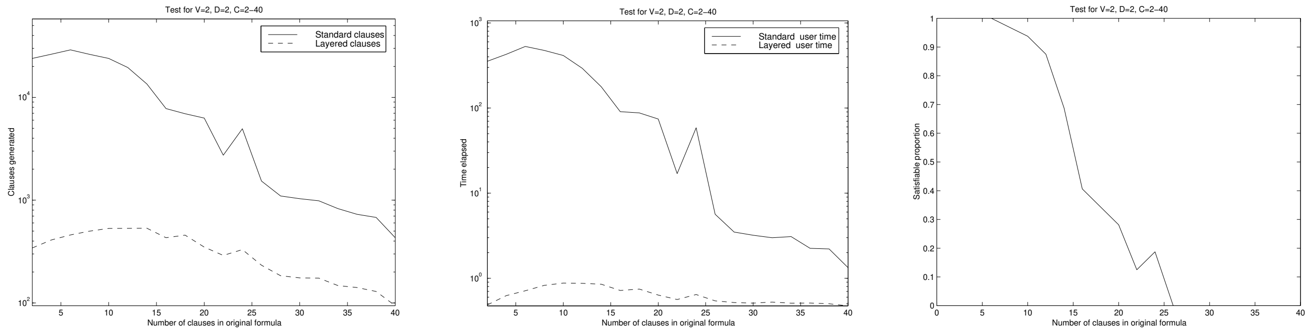


Figure 1. A sample from the tests.

In Figure 1 we display a sample from our experimental results: 64 instances of the 10/3/1 configuration. The top curve indicates the CPU time needed by the standard relational translation, and the bottom one the CPU time needed by the layered translation. Note that the standard translation can be very sensitive to certain hard problems, which results in significant differences between easy and hard instances; the layered method responds in a much more controlled way to hard problems. 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 latter phenomenon can also be observed more globally. The plots in Figure 2 were obtained with V $= $ D $= 2$, while C ranged from 2 to 40. Figures 2 (a) and (b) show the number of clauses generated and the CPU time needed, respectively, for the standard and layered method, while 2 (c) plots the proportion of satisfiable instances as C increases. The curves for the standard and layered methods are very similar, with the layered method lacking the sharp lows and highs that seem to be characteristic for the relational method. Both display a clear easy-hard-easy behavior, but the layered translation is better by several orders of magnitude. Note that the biggest improvements are achieved in the satisfiable region, i.e., for C < 26.

Once we were confident that the layered method consistently displayed a good behavior and a significant improvement over the standard translation, we ran the standardized tests provided by TANCS



(a) (b) (c)

Figure 2. Easy-hard-easy.

(64 instances randomly generated with the 20-clauses/2-variables/2-depth parameters); see Figure 3 for the outcomes.

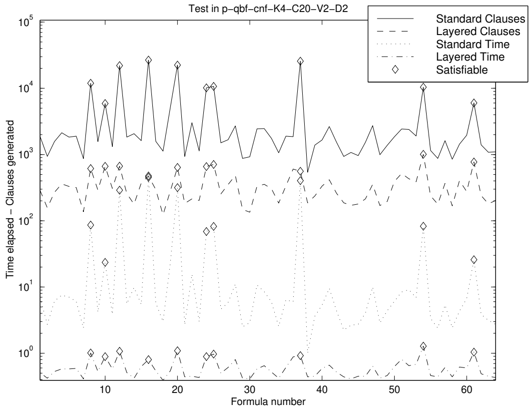


Figure 3. Standard TANCS test 20/2/2.

Finally, to obtain the results in Figure 4 we generated 64 instances of problems for 2 and 3 variables with depths ranging from 1 to 6, again with a time out of 3 hours. The figure shows the average values we obtained. We ran the same tests with the standard instead of the layered translation, but even for moderate depths the computing time and number of clauses exceeded the available resources.

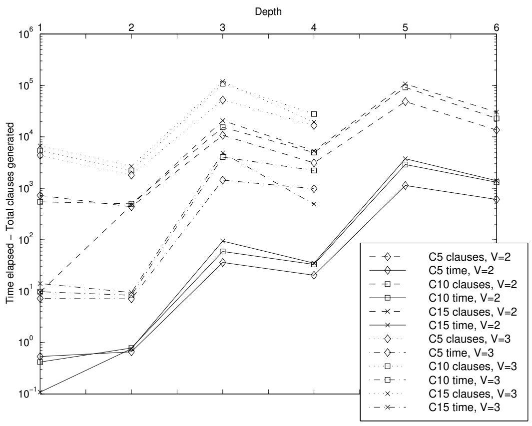


Figure 4.

Additional Tests. Given that the problems generated by the QBF generator were generally too hard for the prover using the standard translation, we also performed tests with a number of 'easier' problem sets, including the one proposed by Heuerding and Schwendimann [7], which were used in, for example, Tableaux'98. Invariably, the layered translation outperformed the standard one; it was able to solve substantially harder instances in all categories.

6 CONCLUSION

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 form of the tree model property in the translation. Using our tree-based heuristics, we have consistently seen improvements, both in terms of the number of clauses generated and in terms of CPU time used. Our ongoing and future work is aimed at exploring the behavior of our heuristics in larger parts of the problem space, 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.

ACKNOWLEDGEMENTS

Juan Heguiabehere and Maarten de Rijke were supported by the Spinoza project 'Logic in Action.' We thank Renate Schmidt for help with SPASS' settings, and the referees for their valuable comments.

REFERENCES

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

[2] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, Cambridge University Press, to appear.

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

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

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

[6] 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).

[7] A. Heuerding and S. Schwendimann, 'A benchmark method for the propositional modal logics K, KT, S4', Report IAM-96-015, University of Bern, (1996).

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

[9] 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, (2000).

[10] SPASS Version 1.0.3. URL: `http://spass.mpi-sb.mpg.de/`. Accessed May 23, 2000.

[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).