Tree-Based Heuristics in Modal Theorem Proving

Carlos Areces Rosella Gennari Juan Heguiabehere
Maarten de Rijke

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

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* ST, which translates modal formulas by transcribing their truth definitions in first-order terms. Let ϕ be a modal formula and x a first-order variable; then $ST_x(\phi)$ is defined as follows:

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

ST commutes with $\wedge$ and $\neg$. In (1), P is a unary predicate symbol corresponding to the proposition letter p; in (2), the variable y is fresh. Observe how (2) reflects the truth definition of the modal operator $\Diamond$.

First-order theorem provers perform poorly on the outputs of this translation, and very sophisticated decision procedures have been developed to overcome this problem. We describe a very intuitive and effective heuristic that can be implemented on top of existing strategies and procedures. 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: **K**-satisfiable) if and only if it is satisfiable at the root of a model based on a tree.

Boosting the relational translation. Our improved relational translation of modal formulas into first-order logic, proceeds in two steps: we will first translate into an intermediate modal language, and from there into first-order logic; the latter step is simply the relational translation ST. The key idea is to label unary and binary relations according to the *number* 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.

Let ϕ be a modal formula and n a natural number. The translation $Tr(\phi, n)$ of ϕ into an intermediate multi-modal language is defined as follows: $Tr(p, n) := p_n$ and $Tr(\Diamond\psi, n) := \Diamond_{n+1} Tr(\psi, n+1)$. The *layered relational translation* is the composition of Tr and ST, and it can be shown to preserve satisfiability.

The layered relational translation provides a new way of turning modal problems into first-order problems. It is conservative in that it can work *on top of* existing strategies. In particular, the layered translation maps modal formulas into the modal fragment, thus existing decision procedure for the modal fragment of first-order logic can still be used.

Experimental results. To evaluate our tree-based heuristics, we used the modal QBF benchmark, the basic yardstick for the TANCS competition on theorem proving and satisfiability testing for non-classical logics. It is a random problem generator designed for evaluating solvers of (un-) satisfiability problems for the modal logic **K**. The formulas of this benchmark are generated using quantified Boolean formulas; a quantified Boolean formula with C clauses is generated with a quantifier alternation depth D, and at most V variables per each alternation. The resul is translated into modal logic using an encoding going back to Halpern.

Tests were performed on a Sun ULTRA II (300MHz) with 1Gb RAM, under Solaris 5.2.5, with SPASS version 1.0.3. The table to the right compares the average time in CPU seconds and number of clauses for two methods: *lay.* (our improved translation) and *stand.* (the relational method). "C/V/D" 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., round$(-1 * \log(N/N'))$. We used 10 samples/point, and a time out of 3 hours on a shared machine; – indicates that a value is not available due to a time out. The layered translation outperformed the standard one in every case, in terms of both CPU time and number of clauses generated; this is not just the average behavior but it was observed in each instance.

C/V/D	Avg. Time		M	Avg. Clauses		M
	Lay.	Stand.		Lay.	Stand.	
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	–	–	10714	–	–
10/2/3	58.996	–	–	15395	–	–
15/2/3	94.192	2094.4	1	20786	45798	0
5/2/4	20.362	–	–	3121	–	–
10/2/4	33.084	–	–	4971	–	–
15/2/4	35.068	–	–	5358	–	–
5/2/5	1136.1	–	–	48546	–	–
10/2/5	2896	–	–	91767	–	–
15/2/5	3758.2	–	–	106870	–	–
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	–	–	1804	–	–
10/3/2	8.3192	–	–	2221	–	–
15/3/2	9.3902	–	–	2687	–	–
5/3/3	1445.2	–	–	52153	–	–
10/3/3	4045.1	–	–	107800	–	–
15/3/3	4865.4	–	–	119150	–	–

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 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 **K4**, **S4**, and temporal logic.