Implementing and Evaluating Provers for First-order Modal Logics

Christoph Benzmüller\textsuperscript{1,2} and Jens Otten\textsuperscript{3} and Thomas Raths\textsuperscript{4,5}

Abstract. While there is a broad literature on the theory of first-order modal logics, little is known about practical reasoning systems for them. This paper presents several implementations of fully automated theorem provers for first-order modal logics based on different proof calculi. Among these calculi are the standard sequent calculus, a prefixed tableau calculus, an embedding into simple type theory, an instance-based method, and a prefixed connection calculus. All implementations are tested and evaluated on the new QML TP problem library for first-order modal logic.

1 Introduction

Modal logics extend classical logic with the modalities "it is necessarily true that" and "it is possibly true that" represented by the unary operators \(\Box\) and \(\Diamond\), respectively. First-order modal logics (FMLs) extend propositional modal logics by domains specifying sets of objects that are associated with each world, and the standard universal and existential quantifiers [6, 9, 10, 13].

FMLs allow a natural and compact knowledge representation. The subtle combination of the modal operators and first-order logic enables specifications of epistemic, dynamic and temporal aspects, and of infinite sets of objects. For this reason, FMLs have many applications, e.g., in planning, natural language processing, program verification, querying knowledge bases, and modeling communication.

All these applications motivate the use of automated theorem proving (ATP) systems for FMLs. Whereas there are some ATP systems available for propositional modal logics, e.g., MSPASS [14] and modleanTAP [1], there were — until recently — no (correct) ATP systems that can deal with the full first-order fragment of modal logics. Relatively little is known about the new ATP systems for FML presented in this paper, in particular, about their underlying calculi and their performance.

The purpose of this paper is to introduce these new ATP systems to the wider AI community and to evaluate and compare their performance. The contributions of this paper include (i) a description of the new ATP systems for FML, (ii) an extension of one of the presented approaches (the simple type theory embedding of FML [4]) is extended from constant domain semantics to varying and cumulative domain semantics), and (iii) an evaluation of these systems exploiting the new QMLTP library, which provides a standardized environment for the application and evaluation of FML ATP systems.

This paper is structured as follows. Section 2 starts with some preliminaries. In Section 3 ATP systems for FML and their underlying proof search calculi are described; these are all sound and available FML ATP systems that exist to date. Section 4 outlines the QMLTP library and infrastructure. Section 5 provides performance results of all described ATP systems. Section 6 concludes the paper.

2 Basics

The syntax of first-order modal logic adopted in this paper is:

\[ F, G ::= \neg F \mid F \land G \mid F \lor G \mid F \rightarrow G \mid \Box F \mid \Diamond F \mid \forall x F \mid \exists x F. \]

The symbols \(P\) are \(n\)-ary \((n \geq 0)\) relation constants which are applied to terms \(t_1, \ldots, t_n\). The \(t_i (0 \leq i \leq n)\) are ordinary first-order terms and they may contain function and constant symbols. Primitive equality is not included (yet); when equality occurs in example problems its properties are explicitly axiomatized. The usual precedence rules for logical constants are assumed. The formula \(\Diamond \exists x P f x \land \Box \forall y (\Diamond y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z\) is used as a running example in this paper, it is referred to as \(F_1\).

The motivation of this paper is practical. Philosophical debates, e.g., the possibilist-actualist debate [11], are deliberately avoided.

Regarding semantics a well accepted and straightforward notion of Kripke style semantics for FML is adopted [9, 13]. In particular, it is assumed that constants and terms are denoting and rigid, i.e. they always pick an object and this pick is the same object in all worlds.

Regarding the universe of discourse constant domain, varying domain and cumulative domain semantics are considered. With respect to these base choices the normal modal logics K, K4, K5, B, D, D4, T, S4, and S5 are studied.

3 Implementations

Sound ATP systems for FML are: the sequent prover MleanSeP, the tableau prover leanTAP, the connection prover MleanCoP, the instance-based method f2p-MSPASS, and modal versions of the higher-order provers LEO-II and Satallax. Table 1 shows for which modal logics these ATP systems can be used.

<table>
<thead>
<tr>
<th>ATP system</th>
<th>modal logics</th>
<th>domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>MleanSeP 1.2</td>
<td>K,K4,D,D4,T,S4</td>
<td>const,cumul</td>
</tr>
<tr>
<td>MleanTAP 1.3</td>
<td>D,T,S4,S5</td>
<td>const,cumul,vary</td>
</tr>
<tr>
<td>MleanCoP 1.2</td>
<td>D,T,S4,S5</td>
<td>const,cumul,vary</td>
</tr>
<tr>
<td>f2p-MSPASS 3.0</td>
<td>K,K4,K3,KB,D,T,S4,S5</td>
<td>const,cumul</td>
</tr>
<tr>
<td>LEO-II 1.3.2-M1.0</td>
<td>K,K4,K5,B,D,D4,T,S4,S5</td>
<td>const,cumul,vary</td>
</tr>
<tr>
<td>Satallax 2.2-M1.0</td>
<td>K,K4,K5,B,D,D4,T,S4,S5</td>
<td>const,cumul,vary</td>
</tr>
</tbody>
</table>

1 FU Berlin, email: c.benzmueller@googlemail.com
2 The author is funded by the German Research Foundation DFG under reference number BE2501/9-1.
3 University of Potsdam, email: jotten@cs.uni-potsdam.de
4 University of Potsdam, email: traths@cs.uni-potsdam.de
5 This author is funded by the German Research Foundation DFG under reference number KR858/9-1.
3.1 Sequent Calculus

The modal sequent calculus extends the classical sequent calculus [12] by the modal rules □-left, □-right, ◦-left, and ◦-right. These rules are used to introduce the modal operators □ and ◦ into the left side or right side of the sequent, respectively [26].

Definition 1 (Modal sequent calculus) The sequent calculus for the modal logics K, K4, D, D4, T, and S4 consists of the axiom and rules of the classical sequent calculus and the four additional rules shown in Figure 1 with Γ := {□G | G ∈ Γ}, Δ0 := {□G | G ∈ Δ}, Γ := {□G | G ∈ Γ}, Δ0 := {□G | G ∈ Δ}, Γ := Γ0 ∪ Γ(3), and Δ0 := Δ0 ∪ Δ(3).

![Figure 1. The additional rules of the (cumulative) modal sequent calculus](image)

MeanSeP is a prover that implements the standard sequent calculus for several modal logics. It is written in Prolog and proof search is carried out in an analytic way. In order to optimize the proof search in the standard calculus of Figure 1, MeanSeP uses free variables and a dynamic Skolemization that is calculated during the proof search. Together with the occurs-check of the term unification algorithm this ensures that the Eigenvariable condition is respected. To deal with constant domains, the Barcan formula (scheme) is automatically added to the given formula in a preprocessing step.

Example 1 (Modal sequent calculus) A derivation of the running example formula F₁ in the modal sequent calculus for the modal logic T (and cumulative domain) is shown in Figure 2.

![Figure 2. A proof for F₁ in the modal sequent calculus](image)

3.2 Tableau Calculus

The classical tableau calculus [21] can be extended to modal logic by adding a prefix to each formula in a tableau derivation [8]. An optimization of this approach uses free variables not only within terms but also within prefixes. It is inspired by the modal matrix characterization of logical validity [26] but uses a tableau-based search to find complementary connections. A prefix is a string consisting of variables and constants, and represents a world path that captures the particular Kripke semantics of the modal logic in question. A prefixed formula has the form F_{pol}^p, where F is a modal formula, pol ∈ {0, 1} is a polarity and p is a prefix.

Definition 2 (Modal tableau calculus) The tableau calculus for the modal logics D, T, S4, and S5 consists of the rules of the classical tableau calculus (which do not change the prefix p of formulae) and the four additional rules shown in Figure 3. V^+ is a new prefix variable, a^+ is a new prefix constant and ◦ is the string concatenation operator. A branch is closed (∗) if it contains a pair of literals of the form {A_1 p_1, A_2 p_2} that are complementary under a term substitution σ_Q and an additional modal substitution σ_M, i.e., σ_Q(A_1) = σ_Q(A_2) and σ_M(p_1) = σ_M(p_2). A tableau proof for a prefixed formula F_{pol}^p is a tableau derivation such that every branch is closed for the pair of substitutions (σ_Q, σ_M). A proof for a modal formula F is a proof for F^↓:ε.

![Figure 3. The four additional rules of the modal tableau calculus](image)

The particular modal logic is specified by distinct properties of the modal substitution σ_M, and an additional admissible criterion on σ_M is used to capture the different domain variants, i.e., constant, cumulative, and varying domain; see Section 3.3 for details.

MeanTAP implements the modal tableau calculus. The compact code is written in Prolog. At first MeanTAP performs a purely classical proof search. After a classical proof is found, the prefixes of the literals that close the branches in the classical tableau are unified. To this end a specialized string unification algorithm is used. If the prefix unification fails, alternative classical proofs (and prefixes) are computed via backtracking. For each modal logic a specific unification algorithm is used that respects the properties and the admissible criterion of the modal substitution for that logic.

Example 2 (Modal tableau calculus) A tableau proof for F₁ with σ_Q(y) = σ_Q(z) = f_0, σ_M(V_1) = σ_M(V_4) = a_1, and σ_M(V_2) = ε (for T, S4) or σ_M(V_2) = a_1 (for S5) is shown in Figure 4.

3.3 Connection Calculus

In contrast to sequent and tableau calculus, which are connective-driven, connection calculi use a connection-driven search strategy. They are already successfully used for automated theorem proving in classical and intuitionistic logic [16, 17]. A connection is a pair of literals, {A, ¬A} or {A_i, A_j}, with the same predicate symbols but different polarities. The connection calculus for classical logic is adapted to modal logic by adding prefixes to all literals and employing a prefix unification algorithm.

---

6 The modal sequent calculus captures the cumulative domain condition. There are no similar cut-free sequent calculi for the logics with constant or varying domain or for the modal logic S5.

7 MeanSeP can be downloaded at [www.leancop.de/mleansep/](http://www.leancop.de/mleansep/).

8 The Barcan formula scheme has the form ∀x(F(x) ⇒ □∀xP(x)) with x = x_1, ..., x_n for all predicates P with n ≥ 1.

9 MeanTAP can be downloaded at [www.leancop.de/mleantap/](http://www.leancop.de/mleantap/).
The prefix of a subformula is defined in the same way as in the tableau calculus (see Section 3.2). Formally, a prefix is a string over an alphabet \( \nu \cup \Pi \), where \( \nu \) is a set of prefix variables, denoted by \( V \) or \( V_i \), and \( \Pi \) is a set of prefix constants, denoted by \( a \) or \( a_i \). Subformulas of the form \( (\sqcap F)^\Pi \) or \( (\sqcup F)^\nu \) extend the prefix by a variable \( V \), subformulas of the form \( (\sqcup F)^\nu \) or \( (\sqcap F)^\nu \) extend the prefix by a constant \( a \) (see also Figure 3), \( \epsilon \) denotes the empty string.

Proof-theoretically, a prefix of a formula \( F \) captures the modal context of \( F \) and specifies the sequence of modal rules of the sequent calculus for each modal logic. A combined substitution \( \sigma \) is a mapping \( \nu \to \nu \cup \Pi \) in \( F \), and the domain condition for the logic \( L \in \{D,S4,S5,T\} \) and the domain \( \Delta \in \{\text{constant},\text{cumulative},\text{varying}\} \) in which all leaves are axioms is called a modal connection proof for \( C, M, Path \in L / D \). A modal connection proof for \( M \) is a modal connection proof for \( \epsilon, M, \epsilon \).

**Theorem 1 (Correctness and completeness)** A (first-order) modal formula \( F \) is valid in the modal logic \( L \) and the domain \( D \) iff there is a modal connection proof for \( M(F) \) in \( L / D \).

The proof of Theorem 1 is based on the the matrix characterization for modal logic [26] and the correctness and completeness of the connection calculus [5]. Proof search in the connection calculus is carried out by applying the rules of the calculus in an analytic way, i.e. from bottom to top. \( \sigma_Q \) and \( \sigma_M \) are calculated by algorithms for term and prefix unification, respectively, whenever a reduction or extension rule is applied. See the work of Otten [18] for details.

**Example 3 (Modal connection calculus)** The prefixed matrix \( M_1 \) of the formula \( F_1 \) from Example 1 is \( \{P^1f : a_1; V_1, V_2, Q^3y : V_1; \} \). A derivation for \( M_1 \) in the modal connection calculus \( \sigma_Q(y) = \sigma_M(z) = \epsilon \) is \( \sigma_Q(V'_1) = \sigma_M(V'_2) = a_1 \) and \( \sigma_M(V'_2) = \epsilon \) (for \( T, S4 \)) or \( \sigma_M(V'_2) = a_1 \) (for \( S5 \)) is shown in Figure 6. \( y', z' \) are \( V'_1, V'_2 \). As all leaves are axioms and the substitution \( \sigma_1 = (\sigma_Q, \sigma_M) \) is admissible the derivation is a proof for \( M_1 \). Hence, the formula \( F_1 \) is valid in the modal logics \( T, S4 \) and \( S5 \).
specific unification algorithm is used for each of the modal logics. The theorem prover for first-order classical logic [16]. To adapt the transfer domain, first2p automatically adds the Barcan formulae (see based ATP system SPASS. It uses several translation methods into clausal form but preserves its structure throughout the whole proof written in Prolog. It does not translate the given formula into any constant. If first2p is unable to add any new instances of subformulae, removes all quantifiers, and replaces every variable with a unique constant. The second component is invoked again in order to add more instances. Afterwards, the propositional ATP system again tries to find a proof or counter model, and so on. This method can be adapted to modal logic by using an ATP system for propositional modal logic. The basic approach works for the cumulative domain and formulae that contain either only existential or only universal quantifiers. This restriction is due to the dependency between applications of modal and quantifier rules, which cannot be captured by the standard Skolemization.

In general, instance-based methods consist of two components. The first component adds instances of subformulae to the given formula and grounds the resulting formula, i.e. removes quantifiers and replaces all variables by a unique constant. The second component uses an ATP system for propositional logic to find a proof or counter model for the ground formula. If a counter model is found, the first component is invoked again in order to add more instances. Afterwards, the propositional ATP system again tries to find a proof or counter model, and so on. This method can be adapted to modal logic by using an ATP system for propositional modal logic. The basic approach works for the cumulative domain and formulae that contain either only existential or only universal quantifiers. This restriction is due to the dependency between applications of modal and quantifier rules, which cannot be captured by the standard Skolemization.

Example 4 (Modal instance-based method) Let $F_1'$ be the modal formula $((\Box Pf \land \Diamond Vy(\Box Py \Rightarrow Qy)) \Rightarrow \exists y Qz$. Initially, the first component of the instance-based method generates the propositional modal formula $((\Box Pf \land \Diamond (\Box Pa \Rightarrow Qa)) \Rightarrow \Diamond (\Box Pa \land Qa)$ by removing all quantifiers and replacing all variables by the unique constant $a$. This formula is refuted by MSPASS and, hence, additional subformula instances are added to $F_1$: $((\Box Pf \land \Diamond (\Box Pa \Rightarrow Qa)) \Rightarrow \Diamond (\Box Pa \land Qa) \land \Diamond Pf)$ and all variables replaced by $a$. Then, the resulting formula $((\Box Pf \land \Diamond (\Box Pa \Rightarrow Qa) \land \Diamond Pf) \Rightarrow \Diamond (Qa \lor Qf))$ is proved by MSPASS.

3.5 Embedding into Classical Higher-Order Logic

Kripke structures can be elegantly modeled in Church’s simple type theory [7], which is also known as classical higher-order logic (HOL). Consequently, prominent non-classical logics, including FMLs, can be encoded as natural fragments of HOL [3].

Definition 4 (Embedding of FML in HOL) Choose HOL type $\iota$ to denote the (non-empty) set of possible worlds and choose an additional base type $\mu$ to denote the (non-empty) set of individuals. As usual, the type $\alpha$ denotes the set of truth values. Certain HOL terms $t_\alpha$ of type $\alpha := \iota \rightarrow \alpha$ then correspond to FML formulae. The logical constants $\neg, \lor, \land, \forall, \exists$, and $\Pi(\forall x F)$ are modeled as abbreviations for the following $\lambda$ terms (types are provided as subscripts):

$$\gamma_{\mu \rightarrow \rho} = \lambda F_{\rho} \lambda w_{\iota} \neg F w$$

$$\Sigma_{\mu \rightarrow \rho} = \lambda F_{\rho} \lambda G_{\rho} \lambda w_{\iota} (F w \lor G w)$$

$$\Pi_{\mu \rightarrow \rho} = \lambda F_{\rho} \lambda w_{\iota} \neg (\exists w F w \lor F w)$$

n-ary relation symbols $P$, n-ary function symbols $f$ and individual constants $c$ obtain types $\mu_1 \rightarrow \ldots \rightarrow \mu_n \rightarrow \rho$, $\rho_1 \rightarrow \ldots \rightarrow \mu_n \rightarrow \mu_{n+1}$ (both with $\mu_i = \mu$ for $0 \leq i \leq n+1$) and $\mu$, respectively. Further logical connectives are defined as usual ($\exists F$ is syntactic sugar for $\Sigma x F$); $\land = \lambda F_{\rho} \lambda G_{\rho} (\neg F \land \neg G)$, $\lor = \lambda F_{\rho} \lambda G_{\rho} (F \lor G)$, $\forall = \lambda F_{\rho} \lambda G_{\rho} (\forall x F x)$, $\exists = \lambda F_{\rho} \lambda G_{\rho} (\exists x F x)$. The constant symbol $R_{\mu \rightarrow \rho}$ denotes the accessibility relation of the $\Box$ operator, which remains unconstrained in logic $K$. For logics $D$, $T$, $S4$, and $S5$, $R$ is axiomatized as serial, reflexive, reflexive and transitive, and an equivalence relation, respectively. This can be done syntactically (e.g. with axiom $\forall x Rx x$ for reflexivity) or syntactically (e.g. with corresponding axiom $\forall x \forall y F x \lor F y$), where quantification over propositions is employed [4]).

Evaluation of a modal formula $F$ for a world $W$ corresponds to evaluating the application $F W$ in HOL. Validity of a modal formula is hence formalized as $\forall W : \mu \forall F_{\rho} F W$. 

Theorem 2 $F$ is a $K$-valid FML formula for constant domain semantics if and only if $\forall F_{\rho} F W$ is valid in HOL for Henkin semantics.

K-valid means validity w.r.t. base modal logic $K$. The theorem follows from Benzmüller and Paulson [4], who study FMLs with quantification over individual and propositional variables (function and constant symbols are avoided there though to achieve a leaner theory).

The ATP systems Satallax and LEO-II are based on Henkin-sound and Henkin-complete calculi for HOL. By Theorem 2 these calculi are also sound and complete for constant domain FMLs.

---

10 MeanCoP can be downloaded at [www.leancoq.de/mleancoq/](http://www.leancoq.de/mleancoq/).

11 For the modal logic K the matrix characterization requires an additional criterion [26], which cannot be integrated into the modal connection calculus or the modal tableau calculus (Section 3.2) in a straightforward way.

12 Arbitrary normal modal logics extending K can be axiomatized this way. However, in some cases only the semantic approach (e.g. for irrellexivity of R) or the syntactic approach (e.g. for McKinsey’s axiom) is applicable.

13 LEO-II can be downloaded from [www.leoprover.org](http://www.leoprover.org), Satallax from [www.ps.uni-saarland.de/~cebrown/satallax/](http://www.ps.uni-saarland.de/~cebrown/satallax/).
LEO-II is based on an extensional higher-order RUE-resolution calculus. It cooperates with a first-order ATP system, by default prover E. Satallax uses a complete ground tableau calculus for higher-order logic to generate successively propositional clauses and calls the SAT solver MiniSat repeatedly to test unsatisfiability of these clauses. It can be regarded as an instance-based method for higher-order logic. Both systems are implemented in OCaml.

Example 5 (Embedding into HOL) Let $F_1^{HOL}$ be the HOL formula $\forall x P f x \land \forall y (Q y \Rightarrow Q y) \Rightarrow \exists z Q z$ for $F_1$ according to Definition 4. The HOL ATP systems are asked to prove $F_1^{HOL}$ instead of $F_1$. The abbreviations of the logical constants are given as equation axioms to the provers, which subsequently ground-expand them. Thus, $F_1^{HOL}$ is rewritten into $\forall w (\neg \forall v (\neg \forall w (\neg R v \lor \neg \forall x (\neg P (f v) \lor \neg \forall v (\neg R v \lor \forall y (\neg \forall v (\neg R v \lor \neg \forall x (\neg P y) \lor Q y)))) \lor \neg \forall v (\neg R v \lor \neg \forall z (Q z)))$. When no further axioms for accessibility relation $R$ are postulated, the ATP systems work for modal logic $K$. In this case, Satallax reports a counter model and LEO-II times out. To adapt the HOL ATP systems e.g. to modal logic $T$, a reflexivity axiom for $R$ is postulated (see above). If respective $T$, $S4$- or $S5$-axioms for $R$ are available then $F_1^{HOL}$ is proved in milliseconds by Satallax and LEO-II. LEO-II delivers a detailed proof object that integrates the contribution of prover $E$ it cooperates with.

As a novel contribution of this paper, the above approach has been adopted for cumulative and varying domain semantics. For this, the following modifications have been implemented:

1. The definition of $\Pi$, which encodes first-order quantification, is modified as follows: $\Pi = \lambda F_\pi, \lambda w, \lambda x, \exists x, \exists w, \text{ExistsInW} \Rightarrow F_\pi w$, where relation $\text{ExistsInW}_{w,v}$ (for ‘$\text{Exists in world}$’) relates individuals with worlds. The sets $\{x \mid \text{ExistsInW} x w\}$ are the possibly varying individual domains associated with the worlds $w$.
2. A non-emptiness axiom for these individual domains is added: $\forall w, \exists x, \exists w, \text{ExistsInW} x w$.
3. For each individual constant symbol $c$ in the proof problem an axiom $\forall w, \exists w, \text{ExistsInW} x w$ is postulated; these axioms enforce the designation of $c$ in the individual domain of each world $w$. Analogous designation axioms are required for function symbols.

Modifications 1–3 adapt the HOL approach to varying domain semantics. For cumulative domain semantics one further modification is needed:

4. The axiom $\forall x, \forall w, \forall w, \exists x, \exists w, \forall w, \text{ExistsInW} x w \land R w \Rightarrow \exists x, \exists w, \exists w, \text{ExistsInW} x w$ is added. It states that the individual domains are increasing along the accessibility relation $R$.

The above approach to automate FMLs in HOL can be employed in combination with any HOL ATP system (however, Satallax and LEO-II are currently the strongest HOL ATP systems [24]). The conversion to the $\text{thf0}$-syntax [22] is realized with the new preprocessor tool $\text{FMLtoHOL}$ (1.0) (hence the suffices ‘-M1.0’ in Table 1).

## 4 The QMLTP Library

The QMLTP library [19] is a benchmark library for testing and evaluating ATP systems for FML, similar to the TPTP library for classical logic [23] and the ILTP library for intuitionistic logic [20]. The most recent version 1.1 of the QMLTP library includes 600 FML problems represented in a standardized extended TPTP syntax divided into 11 problem domains. The problems were taken from different applications, various textbooks, and Gödel’s embedding of intuitionistic logic. It also includes 20 problems in multimodal logic. All problems include a header with many useful information. Furthermore, the QMLTP library includes tools for converting the syntax of FML formulae and provides information of published ATP systems for FML. Further details are provided by Raths and Otten [19].

## 5 Evaluation

The ATP systems presented in Section 3 were evaluated (in automatic mode) on all 580 monomodal problems of version 1.1 of the QMLTP library. The following modal logics were considered: $K$, $D$, $T$, $S4$, and $S5$ with constant, cumulative, and varying domain semantics. Soundness of the provers modulo the problems in the QMLTP library has been checked by comparing the prover results with those of (counter) model finders — some FML ATP systems support both proving theorems and finding (counter) models. Only for GQML-Prover [25] incorrect results have been detected this way and this prover has subsequently been excluded from our experiments.

All tests were conducted on a 3.4 GHz Xeon system with 4 GB RAM running Linux 2.6.24-24.86_64. All ATP systems and components written in Prolog use ECLiPSe Prolog 5.10. LEO II 1.3.2 was compiled with OCaml 3.12 and it works with prover $E$ 1.4. For Satallax a binary of version 2.2 is used. For MSPASS the sources of SPASS 3.0 were compiled using the GNU gcc 4.2.4 compiler. The CPU time limit for all proof attempts was set to 600 seconds.

Table 2 gives an overview of the test results. It contains the number of proved problems for each considered logic and each domain condition for $\text{f2p-MSPASS}$ 3.0, $\text{MleanSeP}$ 1.2, LEO-II 1.3.2-M1.0, Satallax 2.2-M1.0, $\text{MleanTAP}$ 1.3, and $\text{MleanCoP}$ 1.2.

<table>
<thead>
<tr>
<th>Logic/Domain</th>
<th>AT system</th>
<th>F2p-MSPASS</th>
<th>MleanSeP</th>
<th>LEO-II</th>
<th>Satallax</th>
<th>MleanTAP</th>
<th>MleanCoP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$/varying</td>
<td>-</td>
<td>-</td>
<td>72</td>
<td>104</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K$/cumul.</td>
<td>70</td>
<td>121</td>
<td>89</td>
<td>122</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K$/constant</td>
<td>67</td>
<td>124</td>
<td>120</td>
<td>146</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$D$/varying</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td>113</td>
<td>100</td>
<td>179</td>
<td>-</td>
</tr>
<tr>
<td>$D$/cumul.</td>
<td>79</td>
<td>130</td>
<td>100</td>
<td>133</td>
<td>120</td>
<td>200</td>
<td>-</td>
</tr>
<tr>
<td>$D$/constant</td>
<td>76</td>
<td>134</td>
<td>135</td>
<td>160</td>
<td>135</td>
<td>217</td>
<td>-</td>
</tr>
<tr>
<td>$T$/varying</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>170</td>
<td>138</td>
<td>205</td>
<td>224</td>
</tr>
<tr>
<td>$T$/cumul.</td>
<td>105</td>
<td>163</td>
<td>139</td>
<td>192</td>
<td>160</td>
<td>249</td>
<td>224</td>
</tr>
<tr>
<td>$T$/constant</td>
<td>95</td>
<td>166</td>
<td>173</td>
<td>213</td>
<td>175</td>
<td>269</td>
<td>224</td>
</tr>
<tr>
<td>$S4$/varying</td>
<td>-</td>
<td>-</td>
<td>140</td>
<td>207</td>
<td>169</td>
<td>274</td>
<td>-</td>
</tr>
<tr>
<td>$S4$/cumul.</td>
<td>121</td>
<td>197</td>
<td>166</td>
<td>238</td>
<td>205</td>
<td>338</td>
<td>-</td>
</tr>
<tr>
<td>$S4$/constant</td>
<td>111</td>
<td>197</td>
<td>200</td>
<td>261</td>
<td>220</td>
<td>352</td>
<td>-</td>
</tr>
<tr>
<td>$S5$/varying</td>
<td>-</td>
<td>-</td>
<td>169</td>
<td>248</td>
<td>219</td>
<td>359</td>
<td>-</td>
</tr>
<tr>
<td>$S5$/cumul.</td>
<td>140</td>
<td>215</td>
<td>297</td>
<td>272</td>
<td>248</td>
<td>438</td>
<td>-</td>
</tr>
<tr>
<td>$S5$/constant</td>
<td>131</td>
<td>237</td>
<td>305</td>
<td>272</td>
<td>312</td>
<td>438</td>
<td>-</td>
</tr>
</tbody>
</table>

MleanCoP proves the highest number of problems for logics $D$, $T$, $S4$ and $S5$. Satallax comes second for these logics and it performs best for $K$. Satallax and LEO-II have the broadest coverage. $\text{f2p-MSPASS}$ cannot be applied to 299 problems as these problems contain both existential and universal quantifiers (cf. Section 3.4). However, this prover performs particularly well for ‘almost propositional’ formulae, e.g. formulae with a finite Herbrand universe. The

14 The QMLTP library is available online at www.iltp.de/qmltp/.

15 These modal logics are supported by most of the described ATP systems.
The first-order ATP system leanTAP 2.3 [2] was also applied to the 580 problems after removing all modal operators. It prove 296 problems and refutes one.

The new QMLTP problem library has been employed for a first, thorough evaluation of their performance.

Future work includes improvements and extensions of both the first-order modal logic ATP systems and the QMLTP library and related infrastructure. There is obviously a wide spectrum for extensions, including e.g. non-rigid constants and terms, indefinite descriptions, predicate abstractions and multimodal logics.

**Acknowledgements.** We thank Geoff Sutcliffe and Chad Brown for their valuable input to this work.

**REFERENCES**


