Implementing and Evaluating Provers for First-order Modal Logics

Christoph Benzmüller[1 2] and **Jens Otten**[3] and **Thomas Raths**[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 QMLTP 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.

[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: `jeotten@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.

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: $F, G ::= P(t_1, \ldots, t_n) \mid \neg F \mid F \wedge G \mid F \vee G \mid F \Rightarrow G \mid \Box F \mid \Diamond F \mid \forall x F \mid \exists x F$. 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 $(\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$ 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 1. ATP systems for FML

ATP system	modal logics	domains
MleanSeP 1.2	K,K4,D,D4,T,S4	const,cumul
MleanTAP 1.3	D,T,S4,S5	const,cumul,vary
MleanCoP 1.2	D,T,S4,S5	const,cumul,vary
f2p-MSPASS 3.0	K,K4,K5,KB,D,T,S4,S5	const,cumul
LEO-II 1.3.2-M1.0	K,K4,K5,B,D,D4,T,S4,S5	const,cumul,vary
Satallax 2.2-M1.0	K,K4,K5,B,D,D4,T,S4,S5	const,cumul,vary

3.1 Sequent Calculus

The modal sequent calculus extends the classical sequent calculus [12] by the *modal rules* $\Box$*-left*, $\Box$*-right*, $\Diamond$*-left*, and $\Diamond$*-right*. These rules are used to introduce the modal operators $\Box$ and $\Diamond$ into the left side or right side of the sequent, respectively [26].[6]

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* $\Gamma_\Box := \{\Box G \mid \Box G \in \Gamma\}$, $\Delta_\Diamond := \{\Diamond G \mid \Diamond G \in \Delta\}$, $\Gamma_{(\Box)} := \{G \mid \Box G \in \Gamma\}$, $\Delta_{(\Diamond)} := \{G \mid \Diamond G \in \Delta\}$, $\Gamma_{[\Box]} := \Gamma_\Box \cup \Gamma_{(\Box)}$*, and* $\Delta_{[\Diamond]} := \Delta_\Diamond \cup \Delta_{(\Diamond)}$.

$$\frac{\Gamma^+, F \vdash \Delta^+}{\Gamma, \Box F \vdash \Delta}\ \Box\textit{-left} \qquad \frac{\Gamma^* \vdash F, \Delta^*}{\Gamma \vdash \Box F, \Delta}\ \Box\textit{-right}$$

$$\frac{\Gamma^+ \vdash F, \Delta^+}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\textit{-right} \qquad \frac{\Gamma^*, F \vdash \Delta^*}{\Gamma, \Diamond F \vdash \Delta}\ \Diamond\textit{-left}$$

logic	Γ^+	Δ^+	Γ^*	Δ^*	logic	Γ^+	Δ^+	Γ^*	Δ^*
K	(no rules)		$\Gamma_{(\Box)}$	$\Delta_{(\Diamond)}$	D4	$\Gamma_{[\Box]}$	$\Delta_{[\Diamond]}$	$\Gamma_{[\Box]}$	$\Delta_{[\Diamond]}$
K4	(no rules)		$\Gamma_{[\Box]}$	$\Delta_{[\Diamond]}$	T	Γ	Δ	$\Gamma_{(\Box)}$	$\Delta_{(\Diamond)}$
D	$\Gamma_{(\Box)}$	$\Delta_{(\Diamond)}$	$\Gamma_{(\Box)}$	$\Delta_{(\Diamond)}$	S4	Γ	Δ	$\Gamma_\Box$	$\Delta_\Diamond$

Figure 1. The additional rules of the (cumulative) modal sequent calculus

MleanSeP is a prover that implements the standard sequent calculus for several modal logics.[7] 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, MleanSeP 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)*[8] is automatically added to the given formula in a preprocessing step.

Example 1 (Modal sequent calculus) *A derivation of the running example formula F_1 in the modal sequent calculus for the modal logic T (and cumulative domain) is shown in Figure 2.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pfd \vdash Pfd, Qfd}\ \textit{axiom}}{Pfd \vdash \Diamond Pfd, Qfd}\ \Diamond\textit{-right} \qquad \dfrac{}{Pfd, Qfd \vdash Qfd}\ \textit{axiom}}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash Qfd}\ \Rightarrow\textit{-left}}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash \exists z Qz}\ \exists\textit{-right}\ (z \backslash fd)}{Pfd, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z Qz}\ \forall\textit{-left}\ (y \backslash fd)}{\exists x Pfx, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z Qz}\ \exists\textit{-left}\ (x \backslash d)}{\Diamond \exists x Pfx, \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z Qz}\ \Diamond\textit{-left}}{\Diamond \exists x Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z Qz}\ \wedge\textit{-left}}{\vdash (\Diamond \exists x Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz}\ \Rightarrow\textit{-right}$$

Figure 2. A proof for F_1 in the modal sequent calculus

[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] MleanSeP can be downloaded at `www.leancop.de/mleansep/`.

[8] The Barcan formula scheme has the form $\forall \vec{x}(\Box p(\vec{x}) \Rightarrow \Box \forall \vec{x} p(\vec{x})$ with $\vec{x} = x_1, \ldots, x_n$ for all predicates p with $n \geq 1$.

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, $pol \in \{0, 1\}$ 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 $\circ$ is the string concatenation operator. A branch is closed ($\times$) iff it contains a pair of literals of the form $\{A_1^1 : p_1, A_2^0 : p_2\}$ that are complementary under a term substitution σ_Q and an additional modal substitution σ_M, i.e. $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$. 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 $F^0 : \varepsilon$.*

$$\frac{(\Box F)^1 : p}{F^1 : p \circ V^*}\ \Box^1 \qquad \frac{(\Diamond F)^0 : p}{F^0 : p \circ V^*}\ \Diamond^0 \qquad \frac{(\Box F)^0 : p}{F^0 : p \circ a^*}\ \Box^0 \qquad \frac{(\Diamond F)^1 : p}{F^1 : p \circ a^*}\ \Diamond^1$$

Figure 3. The four additional rules of the modal tableau calculus

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.

MleanTAP implements the modal tableau calculus.[9] The compact code is written in Prolog. At first MleanTAP 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 F_1 with $\sigma_Q(y) = \sigma_Q(z) = fd$, $\sigma_M(V_1) = \sigma_M(V_3) = a_1$, and $\sigma_M(V_2) = \varepsilon$ (for T, S4) or $\sigma_M(V_2) = a_1$ (for S5) is shown in Figure 4.*

3.3 Connection Calculus

In contrast to sequent and tableau calculi, 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, $\{A, \neg A\}$ or $\{A^1, A^0\}$, 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.

[9] MleanTAP can be downloaded at `www.leancop.de/mleantap/`.

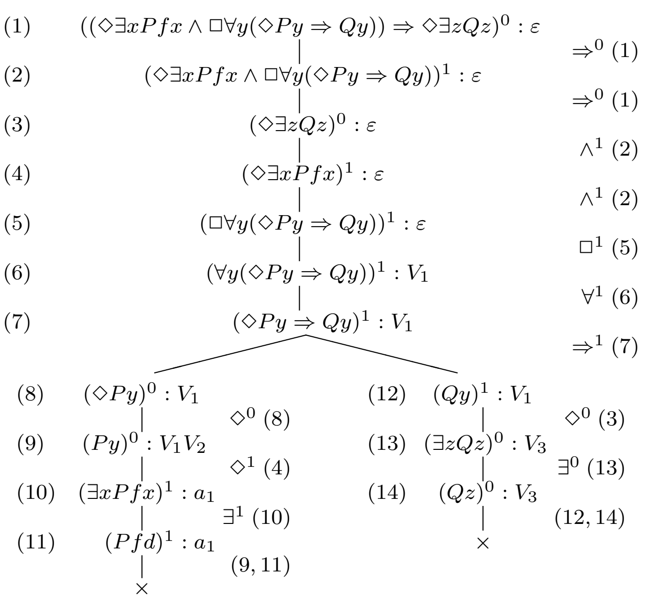
(1) $((\Diamond\exists xPfx \wedge \Box\forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond\exists zQz)^0 : \varepsilon$
$\Rightarrow^0$ (1)
(2) $(\Diamond\exists xPfx \wedge \Box\forall y(\Diamond Py \Rightarrow Qy))^1 : \varepsilon$
$\Rightarrow^0$ (1)
(3) $(\Diamond\exists zQz)^0 : \varepsilon$
$\wedge^1$ (2)
(4) $(\Diamond\exists xPfx)^1 : \varepsilon$
$\wedge^1$ (2)
(5) $(\Box\forall y(\Diamond Py \Rightarrow Qy))^1 : \varepsilon$
$\Box^1$ (5)
(6) $(\forall y(\Diamond Py \Rightarrow Qy))^1 : V_1$
$\forall^1$ (6)
(7) $(\Diamond Py \Rightarrow Qy)^1 : V_1$
$\Rightarrow^1$ (7)
(8) $(\Diamond Py)^0 : V_1$
$\Diamond^0$ (8)
(9) $(Py)^0 : V_1V_2$
$\Diamond^1$ (4)
(10) $(\exists xPfx)^1 : a_1$
$\exists^1$ (10)
(11) $(Pfd)^1 : a_1$
(9, 11)
×
(12) $(Qy)^1 : V_1$
$\Diamond^0$ (3)
(13) $(\exists zQz)^0 : V_3$
$\exists^0$ (13)
(14) $(Qz)^0 : V_3$
(12, 14)
×

Figure 4. A proof for F_1 in the modal tableau calculus

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 $\mathcal{V} \cup \Pi$, where $\mathcal{V}$ is a set of *prefix variables*, denoted by V or V_i, and Π is a set of *prefix constants*, denoted by a or a_i. Subformulae of the form $(\Box F)^1$ or $(\Diamond F)^0$ extend the prefix by a variable V, subformulae of the form $(\Box F)^0$ or $(\Diamond F)^1$ extend the prefix by a constant a (see also Figure 3). ε 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 (see Section 3.1) that have to be applied (analytically) in order to obtain F in the sequent. Semantically, a prefix denotes a specific world in a model [8, 26]. Prefixes of literals that form an axiom in the sequent calculus need to denote the same world, hence, they need to unify under a modal substitution.

A *modal substitution* is a mapping $\sigma_M : \mathcal{V} \to (\mathcal{V} \cup \Pi)^*$ that assigns a string over the alphabet $\mathcal{V} \cup \Pi$ to every element in $\mathcal{V}$. For the modal logics D and T the *accessibility condition* $|\sigma_M(V)| = 1$ or $|\sigma_M(V)| \leq 1$ has to hold for all $V \in \mathcal{V}$, respectively. The accessibility condition encodes the characteristics of the different modal rules in the sequent calculus for each modal logic. A *term substitution* σ_Q is the (usual) mapping from the set of term variables to the set of terms. The substitutions σ_Q and σ_M induce a reduction ordering, which has to be irreflexive [26]. Alternatively, an extended Skolemization technique that was already used for intuitionistic logic [15] can be used for the term Eigenvariables and for the prefix constants. A combined substitution $\sigma := (\sigma_Q, \sigma_M)$ is *admissible* if the following *domain condition* holds for all term variables x and all term variables y occurring in $\sigma_Q(x)$: (i) for cumulative domains $\sigma_M(\mathit{pre}(y)) \preceq \sigma_M(\mathit{pre}(x))$, (ii) for varying domains $\sigma_M(\mathit{pre}(y)) = \sigma_M(\mathit{pre}(x))$. The prefix $\mathit{pre}(x)$ is the prefix of the corresponding subformula $\forall xF$ or $\exists xF$. $u \preceq w$ holds iff u is an initial substring of w or $u = w$.

A connection $\{A_1^1 : p_1, A_2^0 : p_2\}$ is *σ-complementary* if $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$. For the modal logic S5 only the last character of all prefixes is considered (or ε if the prefix is the empty string). The *matrix* of a formula F is a set of clauses that represents the disjunctive normal form of F [5]. In the *prefixed matrix* $M(F)$ of F each literal L is additionally marked with its prefix p.

The modal connection calculus consists of one axiom, a start rule, a reduction rule, and an extension rule. The two latter rules identify σ-complementary connections.

Definition 3 (Modal connection calculus) *The axiom and the rules of the* modal connection calculus *are given in Figure 5. The words of the calculus are tuples of the form "$C, M, Path$", where M is a (prefixed) matrix, C and $Path$ are sets of (prefixed) literals or ε. C is called the* subgoal clause *and $Path$ is called the* active path. *C_1 and C_2 are clauses, $\sigma = (\sigma_Q, \sigma_M)$ is an admissible substitution. σ_Q and σ_M are rigid, i.e. they are applied to the whole derivation.*

$$\textit{Axiom (A)} \quad \frac{}{\{\}, M, Path}$$

$$\textit{Start (S)} \quad \frac{C_2, M, \{\}}{\varepsilon,\ M,\ \varepsilon} \qquad \text{and } C_2 \text{ is copy of } C_1 {\in} M$$

$$\textit{Reduction (R)} \quad \frac{C, M, Path{\cup}\{L_2{:}\,p_2\}}{C{\cup}\{L_1{:}\,p_1\}, M, Path{\cup}\{L_2{:}\,p_2\}}$$

and $\{L_1{:}\,p_1, L_2{:}\,p_2\}$ is σ-complementary

$$\textit{Extension (E)} \quad \frac{C_2 \backslash \{L_2{:}\,p_2\}, M, Path{\cup}\{L_1{:}\,p_1\} \quad C, M, Path}{C{\cup}\{L_1{:}\,p_1\}, M, Path}$$

and C_2 is a copy of $C_1{\in}M$, $L_2{:}p_2 \in C_2$, and $\{L_1{:}\,p_1, L_2{:}\,p_2\}$ is σ-complementary

Figure 5. The modal connection calculus

A derivation for $C, M, Path$ with the admissible substitution $\sigma = (\sigma_Q, \sigma_M)$ that respects the accessibility condition and the domain condition for the logic $\mathcal{L} \in \{$D,S4,S5,T$\}$ and the domain $\mathcal{D} \in \{$constant,cumulative,varying$\}$ in which all leaves are axioms is called a *modal connection proof* for $C, M, Path$ in $\mathcal{L}/\mathcal{D}$. A *modal connection proof* for M is a modal connection proof for $\varepsilon, M, \varepsilon$.

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

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. σ_Q and σ_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 $\{\{P^1 fd : a_1\}, \{P^0 y : V_1V_2, Q^1 y : V_1\}, \{Q^0 z : V_3\}\}$. A derivation for M_1 in the modal connection calculus with $\sigma_Q(y') = \sigma_Q(z') = fd$, $\sigma_M(V_1') = \sigma_M(V_3') = a_1$ and $\sigma_M(V_2') = \varepsilon$ (for T, S4) or $\sigma_M(V_2') = a_1$ (for S5) is shown in Figure 6. y', z' and V_1', V_2', V_3' are new term and prefix variables. The two extension steps use the connections $\{P^1 fd : a_1, P^0 y' : V_1'V_2'\}$ and $\{Q^1 y' : V_1', Q^0 z' : V_3'\}$. 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 $S5$.*

$$\dfrac{\dfrac{\dfrac{}{\{\}, M_1,\{P^1 fd : a_1, Q^1 y' : V_1'\}}A \quad \dfrac{}{\{\}, M_1,\{P^1 fd : a_1\}}A}{\{Q^1 y' : V_1'\}, M_1, \{P^1 fd : a_1\}}E \quad \dfrac{}{\{\}, M_1,\{\}}A}{\{P^1 fd : a_1\}, \{P^0 y : V_1 V_2, Q^1 y : V_1\}, \{Q^0 z : V_3\}\}, \{\}}E}{\varepsilon, \{P^0 y : V_1 V_2, Q^1 y : V_1\}, \{Q^0 z : V_3\}\}, \varepsilon}S$$

Figure 6. A proof for M_1 in the modal connection calculus

MleanCoP [18] is an implementation of the connection calculus for first-order modal logic.[10] It is based on leanCoP, an automated theorem prover for first-order classical logic [16]. To adapt the implementation the leanCoP prover is extended by (a) prefixes that are added to literals and collected during the proof search and (b) a list for each clause that contains term variables and their prefixes in order to check the domain condition. First, MleanCoP performs a classical proof search. After a proof is found, the prefixes of the literals in each connection are unified and the domain condition is checked. A specific unification algorithm is used for each of the modal logics D, T, S4, and S5.[11] The code of the unification algorithm is shared with the unification code of MleanTAP. Furthermore, the following additional techniques that are already used in leanCoP are integrated into MleanCoP: regularity, lemmata, restricted backtracking, a definitional clausal form translation, and a fixed strategy scheduling [17].

3.4 Instance-Based Method

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 quantiers. This restriction is due to the dependency between applications of modal and quantifier rules, which cannot be captured by the standard Skolemization.

f2p-MSPASS implements the instance-based method for various modal logics. It consists of two components. The first component, called first2p, takes a FML formula, adds instances of subformulae, removes all quantifiers, and replaces every variable with a unique constant. If first2p is unable to add any new instances of subformulae, the given FML formula is refuted, i.e. it is not valid. first2p is written in Prolog. It does not translate the given formula into any clausal form but preserves its structure throughout the whole proof process. The second component, MSPASS [14], takes the resulting propositional formula and tries to find a proof or a counter model. MSPASS is an extension of and incorporated into the resolution-based ATP system SPASS. It uses several translation methods into classical logic. By default the standard relational translation from modal logic into classical logic is applied. To deal with the constant domain, first2p automatically adds the Barcan formulae (see Section 3.1) to the given FML formula in a preprocessing step.

[10] MleanCoP can be downloaded at `www.leancop.de/mleancop/`.

[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.

Example 4 (Modal instance-based method) *Let F_1' be the modal formula $(\Diamond Pfd \wedge \Box\forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond\exists z Qz$. Initially, the first component of the instance-based method generates the propositional modal formula $(\Diamond Pfd \wedge \Box(\Diamond Pa \Rightarrow Qa)) \Rightarrow \Diamond Qa$ 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': $(\Diamond Pfd \wedge \Box(\forall y(\Diamond Py \Rightarrow Qy) \wedge (\Diamond Pfd \Rightarrow Qfd))) \Rightarrow \Diamond(\exists z Qz \vee Qfd)$ and all variables replaced by a. Then, the resulting formula $(\Diamond Pfd \wedge \Box((\Diamond Pa \Rightarrow Qa) \wedge (\Diamond Pfd \Rightarrow Qfd))) \Rightarrow \Diamond(Qa \vee Qfd)$ 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 ι to denote the (non-empty) set of possible worlds and choose an additional base type μ to denote the (non-empty) set of individuals. As usual, the type o denotes the set of truth values. Certain HOL terms t_ρ of type $\rho := \iota \to o$ then correspond to FML formulae. The logical constants $\neg, \vee, \Box$, and Π ($\forall x F$ is syntactic sugar for $\Pi\lambda x F$) are modeled as abbreviations for the following λ terms (types are provided as subscripts):*

$$\begin{aligned}
\neg_{\rho\to\rho} &= \lambda F_\rho \lambda w_\iota \neg F w\\
\vee_{\rho\to\rho\to\rho} &= \lambda F_\rho \lambda G_\rho \lambda w_\iota (Fw \vee Gw)\\
\Box_{\rho\to\rho} &= \lambda F_\rho \lambda w_\iota \forall v_\iota (\neg Rwv \vee Fv)\\
\Pi_{(\mu\to\rho)\to\rho} &= \lambda H_{\mu\to\rho} \lambda w_\iota \forall x_\mu H x w
\end{aligned}$$

n-ary relation symbols P, n-ary function symbols f and individual constants c obtain types $\mu_1 \to \ldots \to \mu_n \to \rho$, $\mu_1 \to \ldots \to \mu_n \to \mu_{n+1}$ (both with $\mu_i = \mu$ for $0 \leq i \leq n+1$) and μ, respectively. Further logical connectives are defined as usual ($\exists x F$ is syntactic sugar for $\Sigma\lambda x F$): $\wedge = \lambda F_\rho \lambda G_\rho \neg(\neg F \vee \neg G)$, $\Rightarrow = \lambda F_\rho \lambda G_\rho (\neg F \vee G)$, $\Diamond = \lambda F_\rho \neg\Box\neg F$, $\Sigma = \lambda H_{\mu\to\rho} \neg\Pi\lambda x_\iota \neg H x$. Constant symbol $R_{\iota\to\rho}$ 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 'semantically' (e.g. with axiom $\forall x Rxx$ for reflexivity) or 'syntactically' (e.g. with corresponding axiom $vld\,\forall F_\rho\, \Box F \Rightarrow F$, where quantification over propositions is employed [4]).[12] Evaluation of a modal formula F for a world w corresponds to evaluating the application Fw in HOL. Validity of a modal formula is hence formalized as $vld_{\rho\to o} = \lambda F_\rho \forall w_\iota F w$.

Theorem 2 *F is a K-valid FML formula for constant domain semantics if and only if $vld\, F_\rho$ is valid in HOL for Henkin semantics.*

K-valid means validity wrt. 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.[13] By Theorem 2 these calculi are also sound and complete for constant domain FMLs.

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

[13] LEO-II can be download from `www.leoprover.org`, Satallax from `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 $vld\,((\Diamond\exists x P f x \wedge \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z)$ 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 $\forall w(\neg\neg(\neg\neg\forall v(\neg Rwv \vee \neg\neg\forall x\neg P(fx)v) \vee \neg\forall v(\neg Rwv \vee \forall y(\neg\neg\forall u(\neg Rvu \vee \neg Pyu) \vee Qyv))) \vee \neg\forall v(\neg Rwv \vee \neg\neg\forall z\neg Qzv))$. 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 Π, which encodes first-order quantification, is modified as follows: $\Pi = \lambda F_{\mu\to\rho}\lambda w_\iota \forall x_\mu \text{ExistsInW} x w \Rightarrow F x w$, where relation $\text{ExistsInW}_{\mu\to\iota\to o}$ (for '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: $\forall w_\iota \exists x_\mu \text{ExistsInW} x w$
3. For each individual constant symbol c in the proof problem an axiom $\forall w_\iota \text{ExistsInW} c w$ 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 $\forall x_\mu \forall v_\iota \forall w_\iota \text{ExistsInW} x v \wedge R v w \Rightarrow \text{ExistsInW} x w$ 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 `thf0`-syntax [22] is realized with the new preprocessor tool 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].[14] 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 auto-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.[15] 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.x86_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 f2p-MSPASS 3.0, MleanSeP 1.2, LEO-II 1.3.2-M1.0, Satallax 2.2-M1.0, MleanTAP 1.3, and MleanCoP 1.2.

Table 2. Number of proved monomodal problems of the QMLTP library

Logic/	ATP system					
Domain	f2p-MSPASS	MleanSeP	LEO-II	Satallax	MleanTAP	MleanCoP
K/varying	-	-	72	104	-	-
K/cumul.	70	121	89	122	-	-
K/constant	67	124	120	146	-	-
D/varying	-	-	81	113	100	179
D/cumul.	79	130	100	133	120	200
D/constant	76	134	135	160	135	217
T/varying	-	-	120	170	138	224
T/cumul.	105	163	139	192	160	249
T/constant	95	166	173	213	175	269
S4/varying	-	-	140	207	169	274
S4/cumul.	121	197	166	238	205	338
S4/constant	111	197	200	261	220	352
S5/varying	-	-	169	248	219	359
S5/cumul.	140	-	215	297	272	438
S5/constant	131	-	237	305	272	438

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. 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.

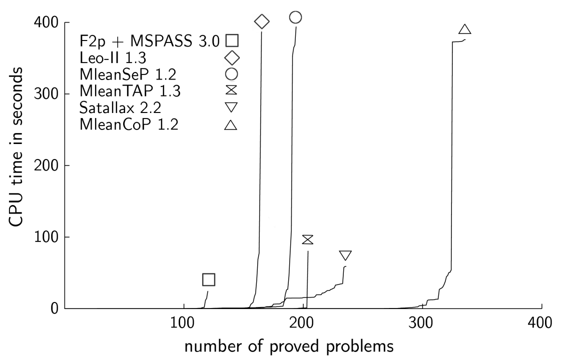

Figure 7. Complexity graph for modal logic S4 with cumulative domains

graph in Figure 7 shows the time complexity behaviour of all FML ATP systems for the logic S4 with cumulative domains. For each prover the associated graph depicts proved problems together with their corresponding solution times (the problems are ordered with respect to their solution time).

f2p-MSPASS, Satallax and MleanCoP also find counter models for many (invalid) FML formulae; e.g. for T with cumulative domains they refute 89, 90, and 125 problems, respectively. Further relevant information is provided in Table 3 and on the QMLTP website.

In addition to the monomodal problems the QMLTP library contains 20 multimodal problems. Currently only LEO-II and Satallax are applicable to those; LEO-II proves 15 problems, Satallax 14.[16]

Table 3. The column entries x/y in this table show (i) the number x of problems that were *exclusively* solved (i.e. proved or refuted) by an ATP system in a particular logic&domain and (ii) the average CPU time y in seconds needed by an ATP system for solving all problems in a particular logic&domain (the full 600s timeout was counted for each failing attempt).

Logic/	ATP system					
Domain	f2p-MSPASS	MleanSeP	LEO-II	Satallax	MleanTAP	MleanCoP
K/varying	-	-	0/529	165/356	-	-
K/cumul.	88/363	4/471	0/511	50/349	-	-
K/constant	42/405	2/471	12/481	45/328	-	-
D/varying	-	-	0/519	0/477	0/492	293/173
D/cumul.	33/407	0/461	0/500	0/464	0/472	194/171
D/constant	33/411	0/462	2/466	0/425	0/456	167/169
T/varying	-	-	0/478	30/320	0/453	121/223
T/cumul.	6/400	0/427	2/456	4/310	0/430	76/217
T/constant	6/410	0/428	2/427	1/295	0/415	66/213
S4/varying	-	-	0/458	30/289	1/421	109/199
S4/cumul.	0/433	0/397	0/430	6/270	1/384	115/163
S4/constant	0/448	0/401	2/397	4/255	1/368	100/162
S5/varying	-	-	0/427	27/265	1/369	132/148
S5/cumul.	0/418	-	0/379	0/244	1/315	126/118
S5/constant	0/436	-	2/359	0/231	1/315	116/118

6 Conclusion

Heterogeneous ATP systems for various first-order modal logics have been presented, including some very recent implementations and some significant modifications of others. These are the first (and currently only) existing, sound ATP systems for FML.

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.

[16] The first-order ATP system leanTAP 2.3 [2] was also applied to the 580 problems after removing all modal operators. It proves 296 problems and refutes one.

REFERENCES

[1] B. Beckert, R. Goré. Free Variable Tableaux for Propositional Modal Logics. In D. Galmiche, Ed., *TABLEAUX-1997*, LNAI 1227, pp. 91–106, Springer, 1997.

[2] B. Beckert, J. Posegga. leanTAP: Lean Tableau-based Deduction. *Journal of Automatic Reasoning*, 15(3): 339–358, 1995.

[3] C. Benzmüller. Combining and Automating Classical and Non-Classical Logics in Classical Higher-Order Logic, *Annals of Mathematics and Artificial Intelligence*, 62:103-128, 2011.

[4] C. Benzmüller, L. Paulson. Quantified Multimodal Logics in Simple Type Theory. *Logica Universalis*, 2012. doi:10.1007/s11787-012-0052-y

[5] W. Bibel. *Automated Theorem Proving*. Vieweg, Wiesbaden, 1987.

[6] P. Blackburn, J. van Benthem, F. Wolter. *Handbook of Modal Logic*. Elsevier, 2006.

[7] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[8] M. Fitting. *Proof Methods for Modal and Intuitionistic Logic*. D. Reidel, Dordrecht, 1983.

[9] M. Fitting, R. L. Mendelsohn. *First-Order Modal Logic*. Kluwer, 1998.

[10] J. Garson. Quantification in Modal Logic. *Handbook of Philosophical Logic*, volume II, pp. 249–307. D. Reidel Publ. Co, 1984.

[11] J. Garson. Unifying Quantified Modal Logic. *Journal of Philosophical Logic*, 34: 621-649, 2005.

[12] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[13] G.E. Hughes, M. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[14] U. Hustadt, R. A. Schmidt. MSPASS: Modal Reasoning by Translation and First-Order Resolution. R. Dyckhoff., Ed., *TABLEAUX-2000*, LNAI 1847, pp. 67–81. Springer, 2000.

[15] J. Otten. Clausal Connection-Based Theorem Proving in Intuitionistic First-Order Logic. In B. Beckert, Ed., *TABLEAUX 2005*, LNAI 3702, pp. 245–261. Springer, 2005.

[16] J. Otten. leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic. In A. Armando, P. Baumgartner, G. Dowek, Eds., *IJCAR 2008*, LNCS 5195, S. 283–291. Springer, 2008.

[17] J. Otten. Restricting Backtracking in Connection Calculi. *AI Communications* 23:159–182, 2010.

[18] J. Otten. Implementing Connection Calculi for First-order Modal Logics. *9th International Workshop on the Implementation of Logics*, Merida/Venezuela, 2012.

[19] T. Raths, J. Otten. The QMLTP Problem Library for First-order Modal Logics. *IJCAR-2012*, LNAI, Springer, 2012. To appear.

[20] T. Raths, J. Otten, C. Kreitz. The ILTP Problem Library for Intuitionistic Logic. *Journal of Automated Reasoning*, 38(1–3): 261–271, 2007.

[21] R. M. Smullyan. *First-Order Logic*. Springer, 1968.

[22] G. Sutcliffe and C. Benzmüller. Automated Reasoning in Higher-Order Logic using the TPTP THF Infrastructure. *Journal of Formalized Reasoning*, 3(1):1-27, 2010.

[23] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[24] G. Sutcliffe. The CADE-23 Automated Theorem Proving System Competition - CASC-23. *AI Communications* 25(1): 49-63, 2012.

[25] V. Thion, S. Cerrito, M. Cialdea Mayer. A General Theorem Prover for Quantified Modal Logics. In U. Egly, C. G. Fermüller, Eds., *TABLEAUX-2002*, LNCS 2381, pp. 266–280. Springer, 2002.

[26] L. Wallen. *Automated deduction in nonclassical logic*. MIT Press, Cambridge, 1990.