Implementing Different Proof Calculi for First-order Modal Logics

– Extended Abstract –

Christoph Benzmüller[1]*, Jens Otten[2] and Thomas Raths[2]†

[1] Institut für Informatik, FU Berlin, Germany
c.benzmueller@googlemail.com
[2] Institut für Informatik, University of Potsdam, Germany
jeotten|traths@cs.uni-potsdam.de

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 [7].

FMLs have many applications, e.g., in planning, natural language processing, program verification, querying knowledge bases, and modeling communication. 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 [9] and modleanTAP [1], there were — until recently — no (correct) ATP systems that can deal with the full first-order fragment of modal logics.

This abstract presents several new ATP systems for FML and sketches their calculi and working principles. The abstract also summarizes the results of a recent comparative evaluation of these new provers (see [4] for further details).

The syntax of first-order modal logic adopted here 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 symbols. The usual precedence rules for logical constants are assumed.

Regarding semantics a well accepted and straightforward notion of Kripke style semantics for FML is adopted [7]. 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, cumulative domain and varying domain semantics are considered.

The following new ATP systems for FML were developed by the authors (partly as extensions of other systems); they support different combinations of modal logics and domain semantics (GQML-Prover [19] has not been included since it returned incorrect results in our experiments for several formulae):

*This author is funded by the German Research Foundation DFG under reference number BE2501/9-1.
†This author is funded by the German Research Foundation DFG under reference number KR858/9-1.

ATP system	base technique	modal logics	domain semantics
MleanSeP 1.2	sequent calculus	K,K4,D,D4,T,S4	const.,cumul.
MleanTAP 1.3	tableau calculus	D,T,S4,S5	const.,cumul.,varying
MleanCoP 1.2	connection calculus	D,T,S4,S5	const.,cumul.,varying
f2p-MSPASS 3.0	instance-based method	K,K4,K5,KB,D,T,S4,S5	const.,cumul.
LEO-II 1.3.2-M1.0	embedding in HOL	K,K4,K5,KB,D,D4,T,S4,S5	const.,cumul.,varying
Satallax 2.2-M1.0	embedding in HOL	K,K4,K5,KB,D,D4,T,S4,S5	const.,cumul.,varying

2 Calculi and ATP Systems for FML

Sequent Calculus. The classical sequent calculus LK [8] is probably the most elegant calculus for classical logic and used in many interactive proof systems. This calculus can be extended to modal logics with cumulative domains by adding the *modal rules* $\Box$*-left*, $\Box$*-right*, $\Diamond$*-left*, and $\Diamond$*-right*. These rules introduce the modal operators $\Box$ and $\Diamond$ into the left side or the right side of the sequent, respectively. All rules of the classical sequent calculus, e.g., the rules for the quantifiers remain unchanged [20].

The *sequent calculus* for the modal logics K, K4, D, D4, T, and S4 with cumulative domains 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 $\Delta_{[\Diamond]} := \Delta_\Diamond \cup \Delta_{(\Diamond)}$. A *sequent proof* for a modal formula F is a derivation of $\vdash F$ in the modal sequent calculus, in which all leaves are closed by axioms.

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

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

logic	Γ^+	Δ^+	Γ^*	Δ^*
D4	$\Gamma_{[\Box]}$	$\Delta_{[\Diamond]}$	$\Gamma_{[\Box]}$	$\Delta_{[\Diamond]}$
T	Γ	Δ	$\Gamma_{(\Box)}$	$\Delta_{(\Diamond)}$
S4	Γ	Δ	$\Gamma_\Box$	$\Delta_\Diamond$

Figure 1: The additional modal rules of the modal sequent calculus

The modal sequent calculus captures the cumulative domain condition. There are no similar cut-free sequent calculi for modal logics with constant or varying domains or for the modal logic S5 [20].

MleanSeP is an ATP system written in PROLOG that implements the sequent calculus for several modal logics. It can be downloaded at `http://www.leancop.de/mleansep/`. MleanSeP performs proof search in an analytic way, i.e. the sequent rules are applied from bottom to top. Furthermore, free-variables are used in combination with a dynamic Skolemization that is calculated during the proof search. Together with the occurs-check of the term unification algorithm this ensures that all derivations respect the Eigenvariable condition. To deal with constant domains, the Barcan formula is automatically added to the given formula in a preprocessing step. 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 $n \geq 1$.

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

Figure 2: The four additional rules of the modal tableau calculus

Tableau Calculus. In general, the (classical) tableau calculus can be seen as compact representations of the (classical) sequent calculus. The classical tableau calculus [16] can be extended to several modal logics by adding a prefix to each formula occurring in a tableau rule [6]. The following tableau calculus for modal logic uses free variables not only within terms but also within prefixes. It is based on the matrix characterization for modal logic [20] but uses a tableau-based search to ensure that all paths contain a complementary connection. A *prefix* is a string consisting of (prefix) variables and (prefix) constants. Essentially, it 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 (first-order) modal formula, *pol* $\in \{0,1\}$ is its polarity and p is its prefix.

The *(prefixed) tableau calculus* for the modal logics D, T, S4, and S5 consists of the rules of the classical tableau calculus [6], which do not change the prefix p of formulae, and the four additional rules shown in Figure 2. V^* is a new prefix variable, a^* is a new prefix constant and $\circ$ is the composition of two strings. A branch is closed if, and only if, 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 all branches are (simultaneously) closed for a pair of term and modal substitutions (σ_Q,σ_M). A *tableau proof* for a modal formula F is a tableau proof for $F^0:\varepsilon$.

In the prefixed tableau calculus the particular modal logic is specified by distinct properties of the modal substitution σ_M. An additional admissible criterion on σ_M is used to capture the different domain variants, i.e., constant, cumulative, or varying domains.

MleanTAP is a compact ATP system written in PROLOG that implements the modal tableau calculus. In can be downloaded at `http://www.leancop.de/mleantap/`. The proof search of MleanTAP is split up into two phases. The first phase performs a purely classical proof search. In the second phase, after a classical tableau proof is found, the prefixes p_1 and p_2 of all literals that close branches in the classical tableau are unified. The unification of these prefixes is done by a specialized string unification algorithm. If the prefix unification fails, alternative classical proofs (and prefixes) are computed. In order to fulfill the distinct properties of the modal substitution σ_M, a specific unification algorithm is used for each modal logic that also respects the admissible criterion.

Connection Calculus. Connection calculi use a *connection-driven* search strategy and are already successfully used for automated theorem proving in classical and intuitionistic logic [11, 12]. 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. 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, and Π is a set of *prefix constants*, denoted by a. It is defined in the same way as in the tableau calculus. 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 2). For the modal logic S5 only the last character of all prefixes is considered (or ε if the prefix is the empty string ε).

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 that have to be applied (analytically) in order to obtain F in the sequent. Semantically, a prefix denotes a specific world in a model [6, 20]. The prefixes of the two literals in a connection, which corresponds to an axiom in the sequent calculus, need to denote

Axiom (A) $\dfrac{}{\{\}, M, Path}$ *Start (S)* $\dfrac{C_2, M, \{\}}{\varepsilon,\ M,\ \varepsilon}$ and C_2 is copy of $C_1 {\in} M$

Reduction (R) $\dfrac{C, M, Path {\cup} \{L_2{:}\,p_2\}}{C {\cup} \{L_1{:}\,p_1\}, M, Path {\cup} \{L_2{:}\,p_2\}}$ and $\{L_1{:}\,p_1, L_2{:}\,p_2\}$ is σ-complementary

Extension (E) $\dfrac{C_2 \backslash \{L_2{:}\,p_2\}, M, Path {\cup} \{L_1{:}\,p_1\} \quad C, M, Path}{C {\cup} \{L_1{:}\,p_1\}, M, Path}$ 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 3: The modal connection calculus

the same world, hence, they need to unify under a modal substitution. A connection $\{A_1^1 : p_1, A_2^0 : p_2\}$ is *σ-complementary*, for $\sigma := (\sigma_Q, \sigma_M)$, if $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$, where σ_Q is the standard *term substitution* and $\sigma_M : \mathcal{V} \to (\mathcal{V} \cup \Pi)^*$ is the *modal substitution* that assigns a string over the alphabet $\mathcal{V} \cup \Pi$ to every element in $\mathcal{V}$. The substitutions σ_Q and σ_M induce a reduction ordering, which has to be irreflexive [20]. Alternatively, a Skolemization technique can be used for the term Eigenvariables and for the prefix constants, as demonstrated by Otten [10].

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 each modal logic. Like for the modal tableau calculus, σ_M has to be admissible with respect to σ_Q. The admissible criterion depends on the domain condition, i.e. it is different for constant, cumulative and varying domains.

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 of F each literal L is additionally marked with its prefix p. The axiom and the rules of the *modal connection calculus* are defined in Figure 3. M is the prefixed matrix of F, the *subgoal clause* C and the *active path* $Path$ are sets of (prefixed) literals or ε. $\sigma = (\sigma_Q, \sigma_M)$ is an admissible substitution and σ_Q and σ_M are rigid, i.e. they are applied to the whole derivation.

A connection proof for $C, M, Path$ is a derivation such that all leaves are axioms for an admissible substitution $\sigma = (\sigma_Q, \sigma_M)$. A *modal connection proof* for the matrix M is a modal connection proof for $\varepsilon, M, \varepsilon$. Correctness and completeness proofs are based on the the matrix characterization for modal logic [20] and the correctness and completeness of the connection calculus [5].

MleanCoP [13] is an implementation of the modal connection calculus. It can be downloaded at `http://www.leancop.de/mleancop/`. It is based on leanCoP, an automated theorem prover for first-order classical logic [11]. To adapt the implementation to the modal connection calculus the leanCoP prover is extended by (a) prefixes that are added to literals and collected during the proof search and (b) an additional list that contains term variables together with their prefixes in order to check the domain condition. First, MleanCoP performs a classical proof search. After a classical proof is found, the prefixes of the literals in each connection are unified and the domain condition is checked. A different unification algorithm is used for each of the modal logics D, T, S4, and S5. For the modal logic K, the matrix characterization [20] requires to check an additional criterion, which cannot be integrated into the modal connection calculus [13] in a straightforward way. This also applies to the modal tableau calculus presented above. MleanCoP uses additional techniques to prune the search space: regularity, lemmata, restricted backtracking, a definition clausal form translation, and a fixed strategy scheduling; see [12] for details.

Instance-Based Method. 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 consists of an ATP system for propositional logic to find a proof or counter model for the ground formula. This method can be adapted to modal logic by using an ATP system for modal propositional logic. The basic approach works for the cumulative domain condition and formulae that contain either only existential or only universal quantifiers. This restriction is due to the dependency of applications of the modal rules and the quantifier rules, which cannot be captured by the standard Skolemization technique.

f2p-MSPASS is an implementation of the instance-based method for first-order modal logic. The first component, called first2p, adds instances of subformulae to the FML formula and grounds the resulting formula. It does not translate the given formula into any clausal form but preserves its structure. For the second component the propositional modal ATP system MSPASS [9] is used. MSPASS is an extension of and incorporated into the resolution-based ATP system SPASS. By default the standard relational translation from modal logic into classical logic is applied. To deal with constant domains, first2p adds the Barcan formula (scheme) to the given FML formula in a preprocessing step.

Embedding into Classical Higher-Order Logic. Various non-classical logics, including FMLs, can be embedded in classical higher-order logic (HOL) [2, 3]. The approach exploits the fact that Kripke structures can be elegantly modeled in HOL [3]: FML propositions F are associated with HOL terms F_ρ of predicate type $\rho := \iota \rightarrow o$. Type o denotes the set of truth values and type ι is associated with the domain of possible worlds. Thus, the application $(F_\rho w_\iota)$ corresponds to the evaluation of FML proposition F in world w. Consequently, validity is formalized as $vld_{\rho\rightarrow o} = \lambda F_\rho \forall w_\iota F w$. Classical connectives like $\neg$ and $\vee$ are simply lifted to type ρ as follows: $\neg_{\rho\rightarrow\rho} = \lambda F_\rho \lambda w_\iota \neg F w$ and $\vee_{\rho\rightarrow\rho\rightarrow\rho} = \lambda F_\rho \lambda G_\rho \lambda w_\iota (F w \vee G w)$. $\Box$ is modeled as $\Box_{\rho\rightarrow\rho} = \lambda F_\rho \lambda w_\iota \forall v_\iota (\neg R w v \vee F v)$, where constant symbol $R_{\iota\rightarrow\iota\rightarrow o}$ denotes the accessibility relation of the $\Box$ operator, which remains unconstrained in logic K. Further logical connectives are defined as usual: $\wedge = \lambda F_\rho \lambda G_\rho \neg(\neg F \vee \neg G)$, $\Rightarrow = \lambda F_\rho \lambda G_\rho (\neg F \vee G)$, $\Diamond = \lambda F_\rho \neg \Box \neg F$. To obtain e.g. modal logic S4, R is axiomatized as reflexive and transitive. Generally, this can be done 'semantically' (e.g. with axiom $\forall x (Rxx)$ for reflexivity) or 'syntactically' (e.g. with axiom $vld\ \forall F_\rho\ \Box F \Rightarrow F$, where quantification over propositions is employed [3]). Arbitrary normal modal logics extending K can be axiomatized this way. However, in some cases only the semantic approach (e.g. for the irreflexivity of R) or the syntactic approach (e.g. for McKinsey's axiom) is applicable.

For individuals a further base type μ is reserved in HOL. Universal quantification $\forall x F$ is introduced as syntactic sugar for $\Pi \lambda x F$, where Π is defined as follows: $\Pi_{(\mu\rightarrow\rho)\rightarrow\rho} = \lambda H_{\mu\rightarrow\rho} \lambda w_\iota \forall x_\mu H x w$. For existential quantification, $\Sigma = \lambda H_{\mu\rightarrow\rho} \neg \Pi \lambda x_\iota \neg H x$ is introduced. $\exists x F$ is then syntactic sugar for $\Sigma \lambda x F$. n-ary relation symbols P, n-ary function symbols f and individual constants c in FML obtain types $\mu_1 \rightarrow \ldots \rightarrow \mu_n \rightarrow \rho$, $\mu_1 \rightarrow \ldots \rightarrow \mu_n \rightarrow \mu_{n+1}$ (with $\mu_i = \mu$ for $0 \leq i \leq n+1$) and μ, respectively.

For any FML formula F holds: F is a valid in modal logic K for constant domain semantics if and only if $vld\, F_\rho$ is valid in HOL for Henkin semantics. This correspondence provides the foundation for proof automation of FMLs with HOL-ATP systems. The correspondence follows from Benzmüller and Paulson [3], who prove a more general result for FMLs with additional quantification over propositional variables. (However, function and constant symbols are avoided in their work to achieve a leaner theory.)

The above approach is adopted for varying domain semantics as follows: 1. Π is now defined as $\Pi = \lambda H_{\mu\rightarrow\rho} \lambda w_\iota \forall x_\mu \texttt{exInW} x w \Rightarrow H x w$, where relation $\texttt{exInW}_{\mu\rightarrow\iota\rightarrow o}$ (for 'exists in world') relates individuals with worlds. 2. The non-emptiness axiom $\forall w_\iota \exists x_\mu \texttt{exInW} x w$ for these individual domains is added. 3. For each individual constant symbol c an axiom $\forall w_\iota \texttt{exInW} c w$ 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.

Table 1: Number of proved monomodal problems of the QMLTP library

Logic	Domain	f2p-MSPASS	MleanSeP	MleanTAP	LEO-II	Satallax	MleanCoP
K	varying	-	-	-	73	104	-
	cumulative	70	121	-	89	122	-
	constant	67	124	-	120	146	-
D	varying	-	-	100	81	113	179
	cumulative	79	130	120	100	133	200
	constant	76	134	135	135	160	217
T	varying	-	-	138	120	170	224
	cumulative	105	163	160	139	192	249
	constant	95	166	175	173	213	269
S4	varying	-	-	169	140	207	274
	cumulative	121	197	205	166	238	338
	constant	111	197	220	200	261	352
S5	varying	-	-	219	169	248	359
	cumulative	140	-	272	215	297	438
	constant	131	-	272	237	305	438

For cumulative domain semantics the axiom $\forall x_\mu \forall v_\iota \forall w_\iota \texttt{exInW} x v \wedge R v w \Rightarrow \texttt{exInW} x w$ is additionally postulated. It states that the individual domains are increasing along relation R.

The above approach can be employed in combination with any HOL ATP system (various candidate systems are presented by Sutcliffe and Benzmüller [17]). Here we use LEO-II (`http://www.leoprover.org`) and Satallax (`http://www.ps.uni-saarland.de/~cebrown/satallax`). The conversion to `thf0`-syntax [17] and the provision of the above axioms is realized with the new preprocessor tool FMLtoHOL (1.0) (hence the suffices '-M1.0' on p. 1).

3 Evaluation Summary

The introduced ATP systems were evaluated on all 580 monomodal problems of version 1.1 of the QMLTP library [14]. The QMLTP library is a benchmark library for testing and evaluating ATP systems for FML, similar to the TPTP library for classical logic [18] and the ILTP library for intuitionistic logic [15]. In the experiments the following modal logics were considered: K, D, T, S4, and S5 with constant, cumulative, and varying domain semantics. These modal logics are supported by most of the described ATP systems. All tests were conducted on a 3.4 GHz Xeon system with 4 GB RAM running Linux 2.6.24-24.x86_64. The CPU time limit was set to 600 seconds. 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 uses 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.

Table 1 gives an overview of the test results for each prover. It contains the number of proved problems for each considered logic and each domain condition. MleanCoP is the strongest prover for logics D, T, S4 and S5, followed by Satallax. For logic K Satallax performs best. f2p-MSPASS cannot be applied to 299 problems as these problems contain both existential and universal quantifiers. f2p-MSPASS, Satallax and MleanCoP also find counter models for many (invalid) FML formulae. E.g., for T with cumulative domains, these ATP systems found counter models for 89, 90, and 125 problems, respectively. In addition to the 580 monomodal logic problems there are also 20 multimodal logic problems in the QMLTP library. Currently, only Satallax and LEO-II are applicable to them. LEO-II proves 15 of these and Satallax 14. The theorem prover leanTAP 2.3 for first-order classical logic was run on the 580 monomodal problems in the QMLTP library, in which all modal operators have been removed. It (classically) proves 296 problems and refutes one problem.

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

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] 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.

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

[4] C. Benzmüller, T. Raths, J. Otten. Implementing and Evaluating Provers for First-order Modal Logics, Proceedings of ECAI'2012. To appear.

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

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

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

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

[9] 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.

[10] 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.

[11] J. Otten. leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic. *IJCAR 2008*, LNCS 5195, pp. 283–291. Springer, 2008.

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

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

[14] T. Raths, J. Otten. The QMLTP Problem Library for First-order Modal Logics. *IJCAR-2012*, to appear.

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

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

[17] 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.

[18] 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.

[19] 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.

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