# MleanCoP: A Connection Prover for First-Order Modal Logic

Jens Otten

Institut für Informatik, University of Potsdam
August-Bebel-Str. 89, 14482 Potsdam-Babelsberg, Germany
jeotten@cs.uni-potsdam.de

**Abstract.** MleanCoP is a fully automated theorem prover for first-order modal logic. The proof search is based on a prefixed connection calculus and an additional prefix unification, which captures the Kripke semantics of different modal logics. MleanCoP is implemented in Prolog and the source code of the core proof search procedure consists only of a few lines. It supports the standard modal logics D, T, S4, and S5 with constant, cumulative, and varying domain conditions. The most recent version also supports heterogeneous multimodal logics and outputs a compact prefixed connection proof. An experimental evaluation shows the strong performance of MleanCoP.

## 1 Introduction

*Modal logics* extend the language of classical logic with the unary modal operators $\Box$ and $\Diamond$. They are used to represent the modalities "it is necessarily true that" and "it is possibly true that", respectively. The *Kripke semantics* of the standard *unimodal* logics are defined by a set of worlds and a single binary accessibility relation between these worlds. *Multimodal* logics consider a finite set of distinct modal operators $\Box_1, \ldots, \Box_n$ and $\Diamond_1, \ldots, \Diamond_n$, and the Kripke semantics is specified by a set of $n$ accessibility relations. *First-order* modal logics extend propositional modal logics by *domains*, i.e. sets of objects that are associated with each world, and the standard universal and existential quantifiers [5,8]. Modal logics have applications in, e.g., planning, natural language processing, and program verification. Multimodal logics are in particular suitable for representing knowledge and beliefs. Popular multimodal logics include temporal and epistemic logic, which are used for program verification and representing dynamic knowledge of different agents [7]. Even though many of these applications would benefit from a higher degree of automation, the development of *efficient* fully automated theorem provers for first-order modal logic is still in its infancy.

This paper presents one of the first theorem provers for first-order (multi)modal logic. It is based on a modal connection calculus (Section 2). Whereas the underlying *connection calculus* provides a basis for an efficient proof search [4,11], *prefixes* are used to directly encode sequences of accessible worlds of the Kripke semantics. The calculus for the different modal logics differ only in the prefix unification, which respects the accessibility relation of the modal logic under consideration. The modal connection calculus is implemented in a very compact Prolog program (Section 3), which shows a strong performance on the problems in the QMLTP library (Section 4).

S. Demri, D. Kapur, and C. Weidenbach (Eds.): IJCAR 2014, LNAI 8562, pp. 269–276, 2014.

## 2 The Modal Connection Calculus

**Syntax and Semantics.** *First-order modal formulae* $F$ are composed of atomic formulae, the standard (classical) connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, the modal operators $\Box$, $\Diamond$, and the standard quantifiers $\forall$ and $\exists$. For *multimodal* logic, sets of modal operators $\{\Box_i, \Diamond_i \mid i \in \mathbb{N}\}$ are considered. The *Kripke semantics* of the standard modal logics are defined by a set of worlds $W$ and a binary *accessibility relation* $R_i \subseteq W \times W$ between these worlds [5,8]. In each single world $w \in W$ the classical semantics applies to the standard connectives and quantifiers, e.g.

$\forall xF/\exists xF$ is true in world $w$ iff $F$ is true in world $w$ for *all/some* object(s) $x$,

whereas the modal operators are interpreted with respect to accessible worlds, i.e.,

$\Box_i F/\Diamond_i F$ is true in world $w$ iff $F$ is true in *all/some* world(s) $w'$, with $(w,w') \in R_i$.

The properties of the accessibility relation $R_i$ determine the particular modal logic. In this paper the modal logics D, T, S4, and S5 are considered. Their accessibility relation is serial (D)[1], reflexive (T), reflexive and transitive (S4), or an equivalence relation (S5). The standard semantics is considered with rigid term designation, i.e. every term denotes the same object in every world, and terms are local, i.e. any ground term denotes an existing object in every world.

**Using Prefixes.** A *prefix* is used to name a sequence of accessible worlds and is assigned to each literal $L$ and each subformula of a given formula $F$. E.g., the prefixed formula $F : w_1 w_2$ denotes the fact that $F$ is true in world $w_2$ that is accessible from a world $w_1$. Similarly to free variables and Skolem terms used for quantified variables, free "world variables" and "Skolem worlds" are used within prefixes [13]. This can be explained by the fact that the semantics of the quantifiers resembles the semantics of the modal operators (see the definitions given above). In the negation normal form $\Box_i$ adds a Skolem world *(prefix constant)* to the prefix, whereas $\Diamond_i$ adds a world variable *(prefix variable)*. Depending on the modal logic (D, T, S4, or S5) and its accessibility relation, variables can be substituted by exactly one prefix variable or constant (D), by at most one prefix variable or constant (T), or by any sequence of prefix variables and constants (S4). For the modal logic S5 only the last element of every prefix is considered.

In Fitting's modal tableau calculi [5,8], prefixes of literals that close a branch need to denote the same world, i.e., they need to be identical. Similarly to term unification for (first-order) terms, this is achieved by a *prefix unification* during the proof search. This unification problem is a special case of string unification that takes the *prefix property*, i.e. the form of the prefixes, and the accessibility relation of the modal logic into account. For D and S5 the prefix unification is straightforward and there is only one most general unifier. For T and S4 prefix unification procedures that calculate minimal (finite) *sets* of most general unifiers were developed as well [10,13].

For (heterogeneous) *multimodal* logics each prefix constant and variable is marked with the index $i$ of the corresponding modal operator $\Box_i$ or $\Diamond_i$. Prefix constants and variables can only be assigned to variables with the same index, and the modal logic assigned to each index $i$ has to be taken into account. Modal operators with different indices are independent from each other, i.e. interaction axioms must be added explicitly.

[1] A relation $R \subseteq W \times W$ is *serial* iff for all $w_1 \in W$ there is some $w_2 \in W$ with $(w_1, w_2) \in R$.

**The Modal Connection Calculus.** The *connection calculus* [4] is already successfully used for automated theorem proving in first-order classical and first-order intuitionistic logic [10,11]. In order to adapt the calculus to modal logic, prefixes are added to all literals. The axiom and the rules of the *modal connection calculus* are given in Figure 1. $M = \{C_1, \ldots, C_m\}$ is a *prefixed matrix*, i.e., a set of clauses where each clause $C_i = \{L_1 : p_1, \ldots, L_n : p_n\}$ is a set of prefixed literals, i.e., $p_i$ is the prefix of the literal $L_i$. The *subgoal clause C* and the *active path Path* are sets of (prefixed) literals or $\varepsilon$; $C_1$ and $C_2$ are clauses. A *connection* $\{L_1 : p_1, L_2 : p_2\}$ is $\sigma$*-complementary* for a *term substitution* $\sigma_Q$ and a *prefix substitution* $\sigma_M$ iff $\sigma_Q(L_1) = \sigma_Q(\overline{L_2})$ and $\sigma_M(p_1) = \sigma_M(p_2)$, where $\overline{L_2}$ is the complement of $L_2$. These substitutions are *rigid*, i.e. they are applied to the whole derivation, and calculated by algorithms for term and prefix unification.

A *modal connection proof* for the prefixed matrix $M$ is a derivation for $\varepsilon, M, \varepsilon$, with admissible substitutions $\sigma_Q$ and $\sigma_M$. Substitutions are *admissible* if they respect the accessibility relation and the domain condition. The accessibility relation depends on the logic and is captured in the specific prefix unification for each modal logic. The *domain condition* ensures that if a Skolem term $t$ is assigned to a variable $x$, then $t$ and $x$ need to exist in the same world. This property holds if the prefix of (the quantifier of) $t$ is an initial string of the prefix of (the quantifier of) $x$ for *cumulative* domains, or if these prefixes are equal for *varying* domains; there is no restriction for *constant* domains.

For example, the prefixed matrix of the modal formula $\Box\forall xPx \Rightarrow \Box\forall y\Box Py$ is $M_1 = \{\{\neg Px : W_1\}, \{Pc : w_2w_3\}\}$ in which $c$ is a Skolem term and $w_2$ and $w_3$ are prefix constants. The following derivation for $M_1$ is a modal connection proof for the modal logics S4 and S5 with constant and cumulative domains (the arc marks the only connection).

$$\dfrac{\dfrac{\overline{\{\},M_1,\{Pc{:}w_2w_3\}}\ \textit{axiom} \qquad \overline{\{\},M_1,\{\}}\ \textit{axiom}}{\{Pc{:}w_2w_3\},\{\{\neg Px{:}W_1\},\{Pc{:}w_2w_3\}\},\{\}}\ \textit{extension}}{\varepsilon,\{\{\neg Px : W_1\},\{Pc : w_2w_3\}\},\varepsilon}\ \textit{start}$$

$\sigma_Q(x) = c$
$\sigma_M(W_1) = w_2w_3$ $(= w_3$ for S5)
(the prefix of $x$ is $w_2w_3$ and the prefix of $c$ is $w_2$)

The modal connection calculus is based on a *clausal matrix characterization* of logical validity [13], which is a slightly adapted version of the original (non-clausal) matrix characterization [15]. In order to simplify the implementation a *Skolemization* is used not only for eigenvariables but also for prefix constants. A similar approach is already used for intuitionistic logic [10]. Thus, the irreflexivity test of the reduction ordering [15] is realized by the occurs check of the term and prefix unification procedures.

$$\textit{axiom}\ \frac{}{\{\},M,Path} \qquad \textit{start}\ \frac{C_2,M,\{\}}{\varepsilon,M,\varepsilon}\ \text{and } C_2 \text{ is copy of } C_1 \in M$$

$$\textit{reduction}\ \frac{C,M,Path\cup\{L_2:p_2\}}{C\cup\{L_1:p_1\},M,Path\cup\{L_2:p_2\}}\ \{L_1:p_1,L_2:p_2\} \text{ is } \sigma\text{-complementary}$$

$$\textit{extension}\ \frac{C_2\setminus\{L_2:p_2\},M,Path\cup\{L_1:p_1\} \quad C,M,Path}{C\cup\{L_1:p_1\},M,Path}\ \begin{array}{l} C_2 \text{ is a copy of } C_1\in M,\ L_2{:}p_2\in C_2,\\ \{L_1{:}p_1,L_2{:}p_2\} \text{ is } \sigma\text{-complementary}\end{array}$$

**Fig. 1.** The connection calculus for first-order modal logic

## 3 The Implementation

MleanCoP implements the modal connection calculus presented in Section 2. Version 1.3 of MleanCoP features the following enhancements compared to version 1.2 [2,13]: support for heterogeneous multimodal logics, output of a compact modal connection proof, support for the modal TPTP syntax, integration of the strategy scheduling into the shell script, and an additional check of the domain condition in the core prover. Furthermore, version 1.3 of MleanCoP does not only support ECLiPSe Prolog, but also SWI and SICStus Prolog. The total size of the shell script and the four files containing the Prolog source code is less than 29 KB. MleanCoP is available under the GNU General Public License and can be downloaded at `http://www.leancop.de/mleancop/` .

**Invoking and Preprocessing.** The MleanCoP prover is invoked by the command

```
./mleancop.sh <problem file> [<time limit>]
```

which starts the proof search for the modal formula in the file `<problem file>`. The optional `<time limit>` is used to control the *fixed strategy scheduling*. If the problem file contains a formula in the modal TPTP syntax [14], it is translated into the MleanCoP syntax. Afterwards, the formula is translated into a prefixed (clausal) matrix, i.e., prefixes are added to all literals in the matrix; *no* other simplifications are carried out in this step. The prefixed matrix is stored in Prolog's database and represented by the predicate `lit/3`. An optional *definitional clausal form* translation reduces the number of possible connections and might prune the search space significantly.

**The Modal Connection Calculus.** The implementation of the core proof search procedure extends the automated theorem prover leanCoP for first-order classical logic [10,11] by adding prefixes to literals and a prefix unification algorithm for each considered modal logic. Furthermore, each clause is annotated with a list that contains term variables together with their prefixes in order to check the domain condition. The Prolog source code of the MleanCoP 1.3 core prover is shown in Figure 2. The underlined code was added to leanCoP 2.1; no other modifications were done. The open subgoal $C$ and the active path *Path* in the modal connection calculus of Figure 1 are represented by the Prolog lists `Cla` and `Path`, respectively. Atoms are represented by Prolog atoms, term (and prefix) variables by Prolog variables and negation by "`-`". The substitutions $\sigma_Q$ and $\sigma_M$ are stored implicitly by Prolog.

The predicate `prove(PathLim,Set,Proof)` (lines *a*–*g*) implements the start rule. `PathLim` is the maximum size of the active path used for *iterative deepening*, `Set` is a list of options used to control the proof search, and `Proof` contains the returned connection proof. First, MleanCoP performs a classical proof search, afterwards, the domain condition is checked ( `domain_cond/1` ) and the collected prefixes are unified ( `prefix_unify/1` ) (line *g*). These are the only external predicates called during the actual proof search. The implementations of the prefix unifications for the modal logics D, T, S4, and S5 need between 2 to 17 lines of Prolog code; the domain condition is implemented by another 15 lines of code. For multimodal logic, prefix constants and variables are marked with the index of the corresponding modal operator. Prefixes are divided into sections and unified according to the modal logic assigned to their indices.

```
(a)  prove(PathLim,Set,Proof) :-
(b)      ( \+member(scut,Set) ->
(c)        prove([(-(#)):(-[])],[],PathLim,[],PreSet,FreeV1,Set,[Proof]) ;
(d)        lit((#):_,FV:C,_) ->
(e)        prove(C,[(-(#)):(-[])],PathLim,[],PreSet,FreeV,Set,Proof1),
(f)        Proof=[C|Proof1], append(FreeV,FV,FreeV1) ),
(g)        domain_cond(FreeV1), prefix_unify(PreSet).
(h)  prove(PathLim,Set,Proof) :-
(i)      member(comp(Limit),Set), PathLim=Limit -> prove(1,[],Proof) ;
(j)      (member(comp(_),Set);retract(pathlim)) ->
(k)      PathLim1 is PathLim+1, prove(PathLim1,Set,Proof).

(1)  prove([],_,_,_,[],[],_,[]).
(2)  prove([Lit:Pre|Cla],Path,PathLim,Lem,[PreSet,FreeV],Set,Proof) :-
(3)      Proof=[[[NegLit:PreN|Cla1]|Proof1]|Proof2],
(4)      \+ (member(LitC,[Lit:Pre|Cla]), member(LitP,Path), LitC==LitP),
(5)      (-NegLit=Lit;-Lit=NegLit) ->
(6)         ( member(LitL,Lem), Lit:Pre==LitL, Cla1=[], Proof1=[],
(7)           PreSet3=[], FreeV3=[]
(8)           ;
(9)           member(NegL:PreN,Path), unify_with_occurs_check(NegL,NegLit),
(10)          Cla1=[], Proof1=[],
(11)          \+ \+ prefix_unify([Pre=PreN]), PreSet3=[Pre=PreN], FreeV3=[]
(12)          ;
(13)          lit(NegLit:PreN,FV:Cla1,Grnd1),
(14)          ( Grnd1=g -> true ; length(Path,K), K<PathLim -> true ;
(15)            \+ pathlim -> assert(pathlim), fail ),
(16)          \+ \+ ( domain_cond(FV), prefix_unify([Pre=PreN]) ),
(17)          prove(Cla1,[Lit:Pre|Path],PathLim,Lem,PreSet1,FreeV1,Set,Proof1),
(18)          PreSet3=[Pre=PreN|PreSet1], append(FreeV1,FV,FreeV3)
(19)        ),
(20)        ( member(cut,Set) -> ! ; true ),
(21)        prove(Cla,Path,PathLim,[Lit:Pre|Lem],PreSet2,FreeV2,Set,Proof2),
(22)        append(PreSet3,PreSet2,PreSet), append(FreeV2,FreeV3,FreeV).
```

**Fig. 2.** Source code of the MleanCoP core prover

The predicate `prove(Cla,Path,PathLim,Lem,[PreSet,FreeV],Set,Proof)` implements the axiom (line 1), the reduction rule (lines 9–11, 21–22) and the extension rule (lines 13, 16–18, 21–22) of the modal connection calculus in Figure 1. A *weak* prefix unification (and domain check) is carried out for the current connection (line 11 and 16); double negation prevents any variable bindings. If the proof search for the current path limit fails and this limit was actually reached (lines 14–15), then `PathLim` is increased and the proof search restarts with an increased path limit (lines *h–k*). MleanCoP uses a few additional effective techniques already used in the classical prover leanCoP: *regularity* (line 4), *lemmata* (lines 6–7), and *restricted backtracking* [11] (line 20). For the example formula from Section 2 the MleanCoP core prover is invoked by

```
prove((# all X: p(X) => # all Y: # p(Y)),Proof).
```

which is (internally) translated into the prefixed matrix

```
[[]:[p(4^[]^[3^[]]):[3^[],5^[]]],[[X,[W]]]:[-(p(X)):-([W])]]
```

and returns the modal prefixed connection proof (for S4 with cumulative domains)

```
Proof = [[p(4^[]^[3^[]]):[3^[], 5^[]]],
          [[-p(4^[]^[3^[]]): -[[3^[], [5^[]]]]]]]
```

where `X` is a term variable, `4^[]^[3^[]]` is a Skolem term; `W` is a prefix variable for the world $W_1$, `3^[]` and `5^[]` are prefix constants for the worlds $w_2$ and $w_3$, respectively.

## 4 Experimental Evaluation

The modal connection prover MleanCoP described in Section 3 was tested on all 580 unimodal and all 20 multimodal problems of version 1.1 of the QMLTP library [14]. All tests were conducted on a 3.4 GHz Xeon system with 4 GB of RAM running Linux 2.6.24 and ECLiPSe Prolog 5.10. The CPU time limit for all proof attempts was set to 100 seconds.

**Table 1.** Results on the unimodal problems (varying/cumul./constant) of the QMLTP library

| Logic | MleanSeP (proved) | MleanTAP (proved) | Satallax (proved) | MleanCoP (proved) | MleanCoP (< 1 sec) | MleanCoP (refuted) |
|---|---|---|---|---|---|---|
| D | – /130/129 | 100/120/135 | 113/133/159 | 186/207/224 | 160/178/193 | 273/247/222 |
| T | – /163/165 | 138/162/175 | 169/192/212 | 223/250/270 | 211/236/253 | 159/132/114 |
| S4 | – /190/189 | 169/205/220 | 206/237/258 | 288/349/364 | 259/304/320 | 127/96/83 |
| S5 | – / – / – | 219/272/272 | 245/294/301 | 359/436/436 | 321/388/388 | 94/41/41 |

Table 1 shows the results for *unimodal* logic for the theorem provers MleanSeP 1.2, MleanTAP 1.3, Satallax 2.2, and MleanCoP 1.3. The columns contain the number of proved problems (proved), and for MleanCoP also the number of problems proved within 1 second (< 1 sec) and the number of refuted problems (refuted). For each logic the results are given for the varying/cumulative/constant domain conditions.

MleanSeP implements the standard modal sequent calculus for several unimodal logics with cumulative domains.[2] It performs an analytic proof search and uses free variables with a dynamic Skolemization. For the constant domain variants the *Barcan formulae* are added. MleanTAP is a compact implementation of a prefixed tableau calculus for several unimodal logics.[3] Similarly to MleanCoP it uses prefixes and an additional prefix unification procedure. Hence, MleanTAP can easily be extended to multimodal logic by integrating the multimodal prefix unification of MleanCoP 1.3. Satallax [6] is a theorem prover for higher-order logic (HOL) and is used in combination with an embedding of first-order modal logic into simple type theory [2,3]. These are currently the only available theorem provers for first-order modal logic. Instead of Satallax, other theorem provers for HOL can be used as well, but Satallax shows the strongest performance when using the embedding into HOL [2].

MleanCoP 1.3 proves significantly more problems than any of the other theorem provers for first-order modal logic. This is true, even if the time limit for MleanCoP is reduced to one second. Satallax comes second, proving more problems than MleanSeP and MleanTAP; it also refutes a high number of problems and can deal with many more modal logics, such as the modal logic K [2].

MleanCoP 1.3 solves 17 of the 20 *multimodal* problems included in the QMLTP library; all of these problems are solved within a fraction of a second.

[2] MleanSeP can be obtained at `http://www.leancop.de/mleansep/`

[3] MleanTAP can be obtained at `http://www.leancop.de/mleantap/`

## 5 Conclusion

Despite the fact that modal logics are considered as some of the most important non-classical logics and numerous calculi were developed, the availability of actual *implementations* of fully automated theorem provers for *first-order* modal logic is very limited. Extending existing theorem provers for *propositional* modal logic, e.g. modleanTAP [1] or MSPASS [9], to *first-order* modal logic is not straightforward [2].

The modal connection calculus extends the classical clausal connection calculus by prefixes and additional prefix unifications, which directly encode the accessibility relations of the different modal logics. MleanCoP is based on the classical connection prover leanCoP and extended by prefix unifications for the unimodal logics D, T, S4, S5 and for the (normal) multimodal logics. The returned modal connection proof contains all necessary information to translate it back into a more readable form.

Future work includes the extension of the classical *non-clausal* connection calculus [12] to first-order modal logic, optimizing the prefix unifications, and extending the prefix unification to other standard modal logics.

## References

1. Beckert, B., Goré, R.: Free Variable Tableaux for Propositional Modal Logics. In: Galmiche, D. (ed.) TABLEAUX 1997. LNCS (LNAI), vol. 1227, pp. 91–106. Springer, Heidelberg (1997)
2. Benzmüller, C., Otten, J., Raths, T.: Implementing and Evaluating Provers for First-order Modal Logics. In: De Raedt, L., et al. (eds.) 20th European Conference on Artificial Intelligence, ECAI 2012, pp. 163–168. IOS Press, Amsterdam (2012)
3. Benzmüller, C., Raths, T.: HOL Based First-Order Modal Logic Provers. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 127–136. Springer, Heidelberg (2013)
4. Bibel, W.: Automated Theorem Proving. Vieweg, Wiesbaden (1987)
5. Blackburn, P., van Bentham, J., Wolter, F.: Handbook of Modal Logic. Elsevier, Amsterdam (2006)
6. Brown, C.: Reducing Higher-Order Theorem Proving to a Sequence of SAT Problems. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 147–161. Springer, Heidelberg (2011)
7. Carnielli, W., Pizzi, C.: Modalities and Multimodalities. Springer, Heidelberg (2008)
8. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Kluwer, Dordrecht (1998)
9. Hustadt, U., Schmidt, R.: MSPASS: Modal Reasoning by Translation and First-Order Resolution. In: Dyckhoff, R. (ed.) TABLEAUX 2000. LNCS (LNAI), vol. 1847, pp. 67–81. Springer, Heidelberg (2000)
10. Otten, J.: leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic. In: Armando, A., Baumgartner, P., Dowek, G., et al. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 283–291. Springer, Heidelberg (2008)
11. Otten, J.: Restricting backtracking in connection calculi. AI Communications 23, 159–182 (2010)
12. Otten, J.: A Non-clausal Connection Calculus. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS (LNAI), vol. 6793, pp. 226–241. Springer, Heidelberg (2011)

13. Otten, J.: Implementing Connection Calculi for First-order Modal Logics. In: Korovin, K., et al. (eds.) IWIL 2012. EPiC, vol. 22, pp. 18–32. EasyChair (2012)
14. Raths, T., Otten, J.: The QMLTP Problem Library for First-order Modal Logics. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 454–461. Springer, Heidelberg (2012)
15. Wallen, L.A.: Automated Deduction in Nonclassical Logics. MIT Press, Cambridge (1990)