Advancing Automated Theorem Proving for the Modal Logics D and S5

Jens Otten

Department of Informatics, University of Oslo, Norway

Abstract

Prefixes encoding the Kripke semantics and an additional prefix unification can be used when performing proof search in some popular non-classical logics. The paper presents two techniques that optimize this approach for the first-order modal logics D and S5. The first one improves backtracking in the modal connection provers MleanCoP and nanoCoP-M, the second one provides an embedding from the modal logics D and S5 into classical first-order logic. Both techniques have been implemented. A practical evaluation on the QMLTP benchmark library shows a significant performance improvement.

Keywords

Automated theorem proving, modal logics, prefix unification, MleanCoP, nanoCoP-M

1. Introduction

Modal logics extend the language of classical logic with the unary modal operators $\Box$ and $\Diamond$ representing the modalities "it is necessarily true that" and "it is possibly true that", respectively. The *Kripke semantics* of the standard modal logics is defined by a set of worlds and a binary accessibility relation between these worlds. Imposing certain restrictions on this relation, such as reflexivity or transitivity, results in different modal logics, such as D, T, S4 or S5 [1].

Some of the most successful automated theorem proving systems for non-classical logics [2, 3, 4] are based on *matrix characterizations* for these logics [5], which use prefixes (or world paths) to capture the Kripke semantics of, e.g., intuitionistic logic or modal logics. *Connection calculi* provide a basis for an efficient proof search [6, 7], while *prefixes* are used to directly encode sequences of accessible worlds of the Kripke semantics. The calculi for the various modal logics differ only in the *prefix unification*, which makes connections complementary and has to take the accessibility relation of the specific modal logic into account.

This paper presents proof search optimization techniques for the modal logics D and S5, which are based on prefixes as used in the matrix characterizations for these logics (Section 2). For these modal logics the prefix unification can be simplified as prefix variables are assigned exactly one character (modal D) or prefixes only consists of at most one character (modal S5). This allows (i) to improve backtracking in the connection provers MleanCoP and nanoCoP, and (ii) an embedding into classical logic for the constant domains (Section 3). These optimizations have been implemented (Section 4) and evaluated on the modal QMLTP library (Section 5).

ARQNL 2022: Automated Reasoning in Quantified Non-Classical Logics, 11 August 2022, Haifa, Israel

jeotten@ifi.uio.no (J. Otten)

https://jens-otten.de (J. Otten)

CEUR Workshop Proceedings (CEUR-WS.org)

2. Prefixes and Matrix Characterization

This section introduces the basic concepts of prefixes and the modal matrix characterization of logical validity for the modal logics D, T, S4 and S5. See [1] for an introduction to modal logics.

2.1. Modal Syntax and Semantics

A *first-order modal formula* F (or G, H) is composed of atomic formulae, the standard connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, the standard quantifiers $\forall$ and $\exists$, and the unary modal operators $\Box$ and $\Diamond$. The *Kripke semantics* of the standard modal logics is defined by a set of worlds W and a binary *accessibility relation* $R \subseteq W \times W$ between these worlds. In each single world $w \in W$ the classical semantics applies to the standard connectives and quantifiers. The modal operators are interpreted with respect to accessible worlds: $\Box F/\Diamond F$ is true in a world w iff F is true in *all/some* world(s) w' with $(w, w') \in R$, i.e., accessible from w.

The properties of the accessibility relation R determine the specific modal logic. For example, the accessibility relation can be serial (modal logic D)[1], reflexive (modal logic T), reflexive and transitive (modal logic S4), or an equivalence relation (modal logic 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.

2.2. Modal Prefixes

A *prefix* is used to name a sequence of accessible worlds in which a formula holds and is assigned to each subformula of a given formula F. For example, the prefix $w_1 \dots w_n$ of a modal formula F denotes that F is true in world w_n that is accessible from a world $w_{n-1}, \dots$, that is accessible from a world w_1. [2]

Definition 1 (Modal prefix). *A* prefix *(denoted by p) is a string (i.e. a sequence of characters) over an alphabet $\nu \cup \Pi$, in which ν is a set of* prefix variables $(V_1, ...)$ *and Π is a set of* prefix constants $(a_1, ...)$*. If a formula F with polarity $pol \in \{0, 1\}$ has the prefix p, written $F^{pol} : p$, then the prefixes (and polarities) of its subformulae G and H are defined inductively according to the table below. It is $\odot \in \{\wedge, \vee\}$, $Q \in \{\forall, \exists\}$; $V \in \nu$ and $a \in \Pi$ are new prefix characters that have not been used before. The* (modal) prefix *of a given (modal) formula F is the empty string ε, the prefixes of its subformulae are the prefixes of the corresponding formula $F^0 : \varepsilon$. The prefix pre(x) of a term variable x occurring in a (sub)formula $Qx\,G$ is the prefix of the subformula G.*

$F^{pol} : p$	$G^{pol'} : p'$	$H^{pol''} : p''$
$(G \odot H)^{pol} : p$	$G^{pol} : p$	$H^{pol} : p$
$(G \Rightarrow H)^{pol} : p$	$G^{1-pol} : p$	$H^{pol} : p$
$(\neg G)^{pol} : p$	$G^{1-pol} : p$	
$(Q\,x\,G)^{pol} : p$	$G^{pol} : p$	

$F^{pol} : p$	$G^{pol'} : p'$
$(\Box G)^1 : p$	$G^1 : pV$
$(\Diamond G)^0 : p$	$G^0 : pV$
$(\Box G)^0 : p$	$G^0 : pa$
$(\Diamond G)^1 : p$	$G^1 : pa$

[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 $(w_1, w_2) \in R$.
[2] The start world, e.g. w_0, is omitted from all prefixes. Therefore, a prefix can also be empty.

Example 1. *Consider the* Barcan formula

$$F_B = (\forall x \,\Box\, p(x)) \Rightarrow (\Box\, \forall y \, p(y)) \,.$$

The prefixes of F_B and its subformulae are: $F_B^0 : \varepsilon$, $(\forall x \,\Box\, p(x))^1 : \varepsilon$, $(\Box\, p(x))^1 : \varepsilon$, $p(x)^1 : V_1$, $(\Box\, \forall y \, p(y))^0 : \varepsilon$, $(\forall y \, p(y))^0 : a_1$, and $p(y)^0 : a_1$, in which V_1 is a new prefix variable and a_1 is a new prefix constant. It is pre$(x) = \varepsilon$ and pre$(y) = a_1$.

Proof-theoretically, prefix variables and constants represent applications of the rules $\Box$-*left*/$\Diamond$-*right* (ν-rules) and $\Box$-*right*/$\Diamond$-*left* (π-rules), respectively, in the modal sequent calculus. A prefix of a formula F specifies the sequence of modal rules that have to be applied (analytically or bottom-up) in order to obtain the formula F in the sequent.

In Fitting's modal tableau calculus [1], prefixes consists only of constant characters and prefixes of literals that close a branch need to denote the same world, i.e., they need to be identical. This is achieved by "guessing the right" sequence of constants by which prefixes are extended whenever a modal ν-rule of the calculus is applied. This "guessing" approach results in a large search space and an inefficient proof search due to necessary backtracking in case the wrong sequence was added.

A more efficient proof search adds a prefix *variable* whenever a modal ν-rule is applied and delays choosing an appropriate sequence of constant characters until a branch is closed. This resembles the approach of using free variables in classical (tableau or sequent) calculi, i.e. using term variables instead of guessing an appropriate first-order term whenever a (non-Eigenvariable) quantifier rule is applied and using term unification whenever a branch is closed in order to assign actual terms to these free term variables. Using prefix variables requires a prefix substitution in order to make prefixes of literals that close a branch identical.

Definition 2 (Prefix substitution). *A* prefix substitution $\sigma_P : \nu \to (\nu \cup \Pi)^*$ *maps elements of ν to strings over $\nu \cup \Pi$. In $\sigma_P(p)$ prefix variables are replaced according to σ_P.*

The accessibility condition and the domain condition ensure that the prefix substitution respects the accessibility relation and the *domain variant* of the considered modal logic.

Depending on the accessibility relation, variables may be substituted by exactly one prefix character (modal logic D), by at most one prefix character (modal logic T) or by an arbitrary string (modal logic S4). For the modal logic S5 only the last character of each prefix is considered (or the the empty string if the prefix is empty).

Definition 3 (Accessibility condition). *For the modal logics D and T the* accessibility condition *is $|\sigma_P(V)|{=}1$ and $|\sigma_P(V)|{\leq}1$ for all $V \in \nu$, respectively.*

Each first-order modal logic can have different domain conditions. If $D(w)$ is the (first-order) domain of a world w, then a modal logic has *constant domains* if $D(w_1) = D(w_2)$ for all $w_1, w_2 \in W$ and it has *cumulative* (or *increasing*) *domains* if $D(w_1) \subseteq D(w_2)$ for all $w_1, w_2 \in W$ with $(w_1, w_2) \in R$. A modal logic has *varying domains* if there are no restrictions on the domains of the worlds.

In case of varying domains, objects may only exists in the world in which they are introduced. For cumulative domains, Eigenvariables[3] have to be introduced before the term variable they are assigned to. For constant domains, there is no such restriction as every object exists in all worlds. The *domain condition* captures these restrictions on prefixes of first-order variables.

Definition 4 (Domain condition). *Let σ_T be a* term *(or* first-order*)* substitution *and σ_P a prefix substitution. The* domain condition *for (a)* varying *and (b)* cumulative *domains states that for all term variables x and all Eigenvariables $\bar{x}$ occurring in the term $\sigma(x)$: (a) $\sigma_P(\text{pre}(\bar{x})) = \sigma_P(\text{pre}(x))$ and (b) $\sigma_P(\text{pre}(\bar{x})) \preceq \sigma_P(\text{pre}(x))$, respectively.[4] For* constant domains *and* S5 with cumulative domains *the domain condition is always true.*

Definition 5 (Admissible substitution). *A* combined substitution *$\sigma = (\sigma_T, \sigma_P)$ is* admissible *with respect to a specific modal logic iff the accessibility condition holds for σ_P and the domain condition holds for σ_T/σ_P for that logic. Furthermore, the* reduction ordering *induced by σ has to be irreflexive (see [5] for details).*

Example 2. *Consider the formula F_B from Example 1. The prefix substitution $\sigma_P(V_1) = a_1$ makes the prefixes of the two atomic formulae $p(x) : V_1$ and $p(y) : a_1$ identical. As $|\sigma_P(V_1)| = 1$, σ_P fulfills the accessibility condition for D, T, S4 and S5. For the term substitution $\sigma_T(x) = y$ it is $\sigma_P(pre(y)) = a_1 \not\preceq \varepsilon = \sigma_P(pre(x))$, hence, $\sigma_B = (\sigma_T, \sigma_P)$ is only admissible for D, T, S4 and S5 with constant domains and for S5 with cumulative domains.*

2.3. Modal Matrix Characterization

A modal formula can be translated into a *prefixed* clausal (disjunctive normal) form, where prefixes are added to atomic formulae. This is called a prefixed matrix. It is a compact representation of the proof search space. The *clausal* modal matrix characterization presented in the following is derived from Wallen's original *non-clausal* characterization of logical validity [5].

Definition 6 (Prefixed matrix and paths). *The* (prefixed) matrix *$M(F)$ representing a modal formula F, is a set of clauses, in which each* clause *is a set of atomic formulae annotated with their prefixes (see [8]). A* path *through a matrix $M = \{C_1, \ldots, C_n\}$ is a set containing one atomic formula from each clause, i.e. $\bigcup_{i=1}^{n} \{A_i\}$ with $A_i \in C_i$ is a path.*

A path represents the set of all atomic formulae in a "final sequent" in the *classical* sequent calculus [9]. Two atomic formulae in a path represent an axiom if they have the same predicate symbol, but different polarities (see Definition 1). They represent an axiom in the *first-order "free-variable" modal* calculus, if their arguments and prefixes unify under a combined substitution.

Definition 7 (Complementary connection). *A* connection *$\{(A_1)^0 : p_1,\ (A_2)^1 : p_2\}$ is a set of atomic formulae with the same predicate symbol but different polarities. A connection is* σ-complementary *for a substitution $\sigma{=}(\sigma_T, \sigma_P)$ iff $\sigma_T(A_1) = \sigma_T(A_2)$ and $\sigma_P(p_1) = \sigma_P(p_2)$.*

[3] *Eigenvariables $\bar{x}$* are exactly those term variables that are quantified in formulae of the form $(\forall \bar{x}\ G)^0$ or $(\exists \bar{x}\ G)^1$.
[4] $u \preceq w$ holds iff u is an initial substring of w or $u = w$.

A *multiplicity* $\mu : M \to \mathbb{N}$ specifies the number of clause copies used in a proof. M^μ denotes the matrix that includes clause copies according to μ. Clause copies correspond to applications of the contraction rule in the sequent calculus.

If (and only if) all paths through a matrix of a formula contain a σ-complementary connection for some admissible substitution σ, the formula is valid. In this case every final sequent of the corresponding derivation in the (free-variable) sequent calculus would contain an axiom (as paths represent final sequents and complementary connections represent axioms).

Theorem 1 (Matrix characterization for modal logics). *A modal formula F is* valid in the modal logic $\mathcal{L}$ *iff there is*

1. *a multiplicity μ,*
2. *a combined substitution $\sigma = (\sigma_T, \sigma_P)$ that is admissible with respect to $\mathcal{L}$, and*
3. *a set $\mathcal{S}$ of σ-complementary connections,*

such that every path through $M^\mu(F)$ contains a connection from $\mathcal{S}$. (See [8, 5] for details.)

Example 3. *Consider the Barcan formula F_B and σ_B from Example 1 and 2. Its prefixed matrix is $M_B{=}M(F_B){=}\{\{p(x)^1{:}V_1\}, \{p(y)^0{:}a_1\}\}$. The only path through M_B is $\{p(x)^1{:}V_1, p(y)^0{:}a_1\}$. It contains the σ_B-complementary connection $\{p(x)^1{:}V_1, p(y)^0{:}a_1\}$, which is (only) admissible for D, T, S4, S5 with constant domains and S5 with cumulative domains. Hence, F_B is (only) valid for the modal logics D, T, S4, S5 with constant domains and S5 with cumulative domains.*

3. Proof Search Optimizations for D and S5

A proof search based on the presented matrix characterization has to calculate an appropriate combined substitution $\sigma{=}(\sigma_T, \sigma_P)$. Whereas the term substitution σ_T is calculated by one of the known algorithms for term unification, calculating the prefix substitution σ_P has to be done by a special prefix unification algorithm [8, 10].

The prefix unification problem is a special case of string unification (the *monoid problem for word equations*) that takes the *prefix property* (or *T-string property*) of the modal prefixes into account: for all prefixes $p_1 = u_1 X w_1$ and $p_2 = u_2 X w_2$ it is $u_1 = u_2$ (with $X \in \nu \cup \Pi$ and $u_1, u_2, w_1, w_2 \in (\nu \cup \Pi)^*$).

In contrast to term unification, which has only a single *most general unifier (mgu)*, the number of mgus for prefix unification might grow exponentially. More specifically, in the worst case the number of mgus is $\frac{1}{2}\frac{(2n)!}{n!)^2} \in \mathcal{O}(\frac{2^{2n}}{\sqrt{n}})$ when unifying the two prefixes $V_1 \ldots V_n$ and $a_1 \ldots a_n$ [11]. For general string unification, the number of mgus might not even be finite, e.g., consider the unification of the two strings Va and aV (with the set of mgus $\{\sigma_P(V) = a^i \mid i \in \mathbb{N}\}$). For the modal logics T and S4, prefix unification procedures that calculate minimal sets of mgus were developed in [11, 8]. In case of the modal logics D the accessibility condition requires $|\sigma_P(V)| = 1$, for the modal logic S5 the prefix is a single character or empty. Hence, for these modal logics the proof search can be optimized as (1) there is only one mgu and (2) the prefix unification can be realized by term unification, allowing an embedding into classical logic.

3.1. Optimizing Prefix Unfication

As the number of mgus for prefix unification can grow, in general, exponentially, it is in practice more efficient to first search for a classical proof, i.e. ignoring all prefixes, and perform the prefix unification and the check of the domain condition afterwards, as pictured below.

1. *classical* proof search: calculate (only) $\mathcal{S}$ and σ_T
2. check *domain condition*: calculate σ'_P
3. *prefix unification*: calculate σ_P (based on σ'_P)

This approach is used by the modal connection provers MleanCoP [3] and nanoCoP-M [4], and also turned out to be successful for intuitionistic provers [12, 4]. But for the modal logics D and S5, for which there is at most one prefix unifier, this leads to redundant backtracking in the classical proof search, which is completely separated from the prefix unification process.

To improve efficiency, the domain condition check and the prefix unification is integrated into the classical proof search, i.e. the prefix substitution σ_P is calculated directly after a connection is identified. This results in a (modal) proof search that consists of only one (major) step instead of three, as pictured below.

1. *modal* proof search: calculate $\mathcal{S}$, σ_T *and* σ_P (simultaneously)

This approach ensures that the calculated set of connections is always σ-complementary and a failed check of the domain condition or prefix unification is quickly detected. Even though this also leads to some overhead, in general, this results in a more efficient proof search for the modal logics D and S5.

Example 4. *Consider the formula* $(\forall x \Diamond p(x)) \Rightarrow (\Box p(b_1) \vee \ldots \vee \Box p(b_n) \vee \Diamond p(c))$ *with the matrix* $\{\{p(x)^0\colon a\}, \{p(b_1)^1\colon a_1\}, \ldots \{p(b_n)^1\colon a_n\}, \{p(c)^1\colon V_1\}\}$. *So far,* MleanCoP *and* nanoCoP-M *first perform a (purely) classical proof search by identifying the connection* $\mathcal{S} = \{p(x)^0\colon a, p(b_1)^1\colon a_1\}$ *with the term substitution* $\sigma_T(x) = b_1$. *But the prefix unification of the two prefixes* a *and* a_1 *fails (the domain condition is always true). Afterwards, backtracking within the classical proof search is done. This results in the next classical proof with* $\mathcal{S} = \{p(x)^0\colon a, p(b_2)^1\colon a_2\}$ *and* $\sigma_T(x) = b_2$. *Again, the prefix unification of* a *and* a_2 *fails. Backtracking within the classical proof continues and results in, altogether,* n *different classical proofs for which the subsequent prefix unification fails. Only for the last possible classical proof with* $\mathcal{S} = \{p(x)^0\colon a, p(c)^1\colon V_1\}$ *and* $\sigma_T(x) = c$ *the prefix unification of* a *and* V_1 *succeeds with* $\sigma_P(V_1) = a$. *An improved procedure that calculates the prefix substitution directly after a connection is identified would immediately identify the modal proof with* $\mathcal{S} = \{p(x)^0\colon a, p^1(c)\colon V_1\}$, $\sigma_T(x) = c$ *and* $\sigma_P(V_1) = a$, *avoiding any backtracking within the classical proof search. That this approach is more efficient becomes clearer for slightly larger formulae with a larger classical proof (search) and, hence, more choices for backtracking.*

3.2. Embedding into Classical Logic

As $|\sigma(V)| = 1$ for the modal logic D and $|p| \leq 1$ for all prefixes p in S5, the prefixes of atomic formulae can be included in their arguments and unified by the standard term unification. This yields an embedding for these modal logics into classical logic.

Definition 8 (Translation into classical logic). *The* translation $\mathcal{T}$ of a prefixed modal formula *into a classical formula is defined according to Table 1. In this table,* $P(\ldots)$ *is an atomic formula,* $t_1, \ldots, t_n$ *are (first-order) terms,* $pol \in \{0, 1\}$ *is a polarity,* $\mathcal{V}$ *is a set of term and prefix variables,* p *and* $c_1 \ldots c_m$ *are prefixes.* f^m_{pre} *is a new function symbol with arity* $m \geq 1$*,* eps_{pre} *is a unique constant,* $V \in \nu$ *is a new prefix variable, and* $c^{|\mathcal{V}|}_{sk}(\mathcal{V})$ *is a new (skolemized) prefix constant parametrized with the variables in* $\mathcal{V}$*. The* translation of a modal formula F *is* $\mathcal{T}(F) := \mathcal{T}(F : \epsilon, \{\})$.

Table 1
The translation $\mathcal{T}$ from the first-order modal logics D and S5 to first-order classical logic

$\mathcal{T}(P(t_1, \ldots, t_n)^{pol} : \varepsilon, \mathcal{V}) = P(t_1, \ldots, t_n, f^1_{pre}(\mathsf{eps}_{pre}))$		for D and S5
$\mathcal{T}(P(t_1, \ldots, t_n)^{pol} : c_1 \ldots c_m, \mathcal{V}) = P(t_1, \ldots, t_n, f^1_{pre}(c_m))$		only for S5
$\mathcal{T}(P(t_1, \ldots, t_n)^{pol} : c_1 \ldots c_m, \mathcal{V}) = P(t_1, \ldots, t_n, f^m_{pre}(c_1, \ldots, c_m))$		only for D

$\mathcal{T}((G \odot H)^{pol} : p, \mathcal{V}) = (\mathcal{T}(G^{pol} : p, \mathcal{V}) \odot \mathcal{T}(H^{pol} : p, \mathcal{V}))$		$\odot \in \{\wedge, \vee\}$
$\mathcal{T}((G \Rightarrow H)^{pol} : p, \mathcal{V}) = (\mathcal{T}(G^{1-pol} : p, \mathcal{V}) \Rightarrow \mathcal{T}(H^{pol} : p, \mathcal{V}))$		
$\mathcal{T}((\neg G)^{pol} : p, \mathcal{V}) = (\neg\, \mathcal{T}(G^{1-pol} : p, \mathcal{V}))$		

$\mathcal{T}((\forall x\, G)^1 : p, \mathcal{V}) = (\forall x\, \mathcal{T}(G^1 : p, \mathcal{V} \cup \{x\}))$	$\mathcal{T}((\Box G)^1 : p, \mathcal{V}) = (\forall V\, \mathcal{T}(G^1 : p\, V, \mathcal{V} \cup \{V\}))$
$\mathcal{T}((\exists x\, G)^0 : p, \mathcal{V}) = (\exists x\, \mathcal{T}(G^0 : p, \mathcal{V} \cup \{x\}))$	$\mathcal{T}((\Diamond G)^0 : p, \mathcal{V}) = (\exists V\, \mathcal{T}(G^0 : p\, V, \mathcal{V} \cup \{V\}))$
$\mathcal{T}((\forall x\, G)^0 : p, \mathcal{V}) = (\forall x\, \mathcal{T}(G^0 : p, \mathcal{V}))$	$\mathcal{T}((\Box G)^0 : p, \mathcal{V}) = \mathcal{T}(G^0 : p\, c^{\lvert\mathcal{V}\rvert}_{sk}(\mathcal{V}), \mathcal{V})$
$\mathcal{T}((\exists x\, G)^1 : p, \mathcal{V}) = (\exists x\, \mathcal{T}(G^1 : p, \mathcal{V}))$	$\mathcal{T}((\Diamond G)^1 : p, \mathcal{V}) = \mathcal{T}(G^1 : p\, c^{\lvert\mathcal{V}\rvert}_{sk}(\mathcal{V}), \mathcal{V})$

The prefixes of atomic formulae are now part of the function f_{pre}, which is added as last argument to each atomic formula. *Skolemization* is extended to the prefix constants (function c_{sk}) in order to check the irreflexivity of the reduction ordering by the occurs check of the term unification.[5] This embedding works for the modal logics that do not have any domain condition, i.e., for D and S5 with constant domains (and, hence, also for S5 with cumulative domains).

Theorem 2 (Correctness of translation $\mathcal{T}$). *A modal formula* F *is valid in the modal logics D or S5 with constant domains iff its translated formula* $\mathcal{T}(F)$ *is valid in classical logic.*

This theorem follows from the matrix characterization (Theorem 1) and the restriction of the accessibility condition for D and S5. The translation $\mathcal{T}$ preserves the structure of the formula, only the modal operators are eliminated and one argument is added to all atomic formulae.

Example 5. *Consider again the Barcan formula*

$$F_B = (\forall x \Box p(x)) \Rightarrow (\Box \forall y\, p(y))\,.$$

Its translation (for both D and S5) is $\mathcal{T}(F_B) = (\forall x \forall V\, p(x, f^1_{pre}(V) \Rightarrow (\forall y\, p(y, f^1_{pre}(c_{sk}))))$*. As* $\mathcal{T}(F_B)$ *is classically valid,* F_B *is valid for the modal logics D and S5 with constant domains (and also for the modal logic S5 with cumulative domains).*

[5] This proof-theoretical view on Skolemization is crucial for its extension to prefixes, see also [6, 12, 8].

4. The Implementations

The optimization and embedding of Section 3 have been implemented within three systems for the first-order modal logics D and S5.

1. MleanCoP-DS5 is an optimized version of the clausal connection prover MleanCoP,
2. nanoCoP-M-DS5 is an optimized version of the non-clausal prover nanoCoP-M,
3. nano-M2C implements an embedding of modal logic into classical logic.

4.1. The Modal Provers MleanCoP-DS5 and nanoCoP-M-DS5

The optimization described in Section 3.1 was integrated into the provers MleanCoP 1.3 [3] and nanoCoP-M 2.0 [4]. MleanCoP and nanoCoP-M are very compact Prolog implementations of the clausal and non-clausal *modal connection calculus*, respectively. They are based on the modal matrix characterization of logical validity as specified in Theorem 1 in Section 2.3. In connection calculi the proof search is guided by connections in order to calculate a set $\mathcal{S}$ of σ-complementary connections [13, 6].

In the original MleanCoP and nanoCoP-M provers the (rigid) prefix substitution σ_P is calculated only after a classical proof (and an appropriate set of connections $\mathcal{S}$) is found. In the optimized "DS5" versions, the prefix substitution σ_P is calculated/extended directly after a new connection is identified. This is done by unifying the prefixes of each new connection and checking the appropriate domain condition. Since there is only a single most general prefix unifier for D and S5, no backtracking over alternative prefix substitutions is necessary.

The optimized provers MleanCoP-DS5 and nanoCoP-M-DS5 support the first-order modal logics D and S5 with constant, cumulative and varying domains.[6]

4.2. The Translation nano-M2C into Classical Logic

The translation $\mathcal{T}$ introduced in Definition 8 of Section 3.2 was implemented by the compact Prolog program nano-M2C. The translation takes a modal formula/problem in QMLTP syntax as input and translates it into a validity preserving formula in classical first-order logic according to Table 1. The translated formula is written into a file and can be further processed by any classical first-order prover, such as leanCoP [14, 2], nanoCoP [15] or E [16].

The translation nano-M2C supports the first-order modal logics D with constant domains and S5 with constant and cumulative domains.[7]

Example 6. *The Barcan formula* $(\forall x \Box p(x)) \Rightarrow (\Box \forall y\, p(y))$ *from Example 1 is included in the QMLTP library [17] as problem* `SYM001+1.p` *where it is presented in the following way:*

```
qmf(con,conjecture,
    (( ! [X] : (#box : ( f(X) ) ) ) => (#box : ( ! [X] : ( f(X) ) )) ) ).
```

It is translated by nano-M2C *into the following classical formula (*`pr`/`c_sko` *are new symbols):*

```
fof(con, conjecture,
    (  ! [X] : ! [V] : f(X, pre(V)) =>   ! [X] : f(X, pre(c_sko(1))) ) ).
```

[6] MleanCoP-DS5/nanoCoP-M-DS5 are available at http://leancop.de/mleancop/ and http://leancop.de/nanocop-m/.
[7] nano-M2C is available at http://leancop.de/nano-m2c/.

5. Experimental Evaluation

The three new implementations MleanCoP-DS5, nanoCoP-M-DS5 and nano-M2Cdescribed in Section 4 were evaluated on all 580 unimodal problems of the QMLTP library v1.1 [17].

All evaluations were conducted on a LIFEBOOK U9311 with a 4-core i7-1185G7 CPU and 16GB of RAM, running Linux 5.13.0. ECLiPSe Prolog version 5.10[8] was used for all Prolog provers and the E prover was compiled with GNU gcc version 9.4.0. The time limit for all test runs was set to 100 seconds.

5.1. MleanCoP-DS5 and nanoCoP-M-DS5

Table 2 shows the results for the modal clausal connection prover MleanCoP 1.3 [3] and its optimized version MleanCoP-DS5 1.0, for the modal non-clausal connection prover nanoCoP-M 2.0 [4] and its optimized version nanoCoP-M-DS5 1.0. The total number of solved problems and the number of proved (valid) and refuted (invalid) problems is given.

Compared to the non-optimized versions, the optimized "'DS5" connection provers solve between 21 and 38 more problems for the modal logic D. For the modal logic S5 the improvement is less significant, the optimized versions solve between two and seven more problems.

Table 2
Results of the modal connection provers on the QMLTP library [total (proved/refuted)]

Logic	MleanCoP	MleanCoP-DS5	nanoCoP-M	nanoCoP-M-DS5
D varying	458 (185/273)	**487** (195/292)	451 (185/266)	**489** (193/296)
D cumulative	453 (206/247)	**474** (215/259)	451 (205/246)	**484** (213/271)
D constant	445 (223/222)	**466** (232/234)	451 (222/229)	**472** (230/242)
S5 varying	454 (360/94)	**458** (360/98)	457 (365/92)	**464** (367/97)
S5 cumulative	477 (436/41)	**483** (441/42)	484 (440/44)	**486** (441/45)
S5 constant	477 (436/41)	**483** (441/42)	484 (440/44)	**486** (441/45)

5.2. nano-M2C

Table 3 shows the results for the nano-M2C embedding into classical logic combined with the classical provers leanCoP 2.1 [2, 7], nanoCoP 2.0 [4], E 2.6 [16] with the options `-s --satauto --proof-object --cpu-limit=100`, and E 2.6 "sched" with the *strategy scheduling* option `--satauto-schedule` instead of `--satauto`. Again, the total number of solved problems and the number of proved (valid) and refuted (invalid) problems is given.

The results using nano-M2C in combination with the classical connection provers leanCoP and nanoCoP are very similar to the optimized "DS5" versions of the modal connection provers. In combination with the E "sched" prover, the nano-M2C translation solves between 44 and 67 more problems than MleanCoP and nanoCoP-M.[9]

[8] ECLiPSe Prolog 5.x is available at https://eclipseclp.org/Distribution/Builds/. All newer versions of ECLiPSe Prolog are missing important features and have a significantly lower performance.
[9] Surprisingly, the non-strategy-scheduling version of E proves less problems than the classical connection provers.

Table 3
Results of the nano-M2C translation on the QMLTP library [total (proved/refuted)]

Logic	M2C+leanCoP	M2C+nanoCoP	M2C+E	M2C+E(sched)
D constant	461 (234/227)	472 (230/242)	500 (229/271)	**512** (234/278)
S5 cumulative	485 (443/42)	486 (441/45)	517 (435/82)	**528** (446/82)
S5 constant	485 (443/42)	486 (441/45)	517 (435/82)	**528** (446/82)

6. Conclusion

This paper presents two techniques for advancing automated theorem proving for the first-order modal logics D and S5. It introduces an optimization of the prefix unification for two of the fastest existing modal provers, MleanCoP and nanoCoP-M, and an embedding of the modal logics D and S5 into classical logic.

Experiences from the development of nanoCoP-M have shown that it is getting increasingly difficult to improve the performance of modal theorem provers on the QMLTP libray [10, 4]. Nevertheless, the two new optimized provers MleanCoP-DS5 and nanoCoP-M-DS5, and the implementation nano-M2C of the embedding into classical logic solve significantly more problems than the MleanCoP and nanoCoP-M provers. The improvement is rather small for S5 with varying domains, but more significant for D with varying and cumulative domains (MleanCoP-DS5 and nanoCoP-M-DS5) and for D and S5 with constant domains and S5 with cumulative domains (nano-M2C in combination with the E prover).

Ohlbach uses so-called *world path* to deal with the Kripke semantics of different modal logics [18, 19]. His approach is very similar to Wallen's matrix characterization on which the work in this paper is based. Ohlbach has based his work on the resolution calculus and uses a less refined prefix unification procedure. Up to the author's knowledge, Ohlbach's procedure has never been implemented. The higher-order prover Leo-III uses the relational translation (modeling the Kripke semantics) of first-order modal logic into typed higher-order logic to deal with a wide range of modal logics [20].[10]

Future work includes improvements of the prefix unification algorithm for some other modal logics, such as T and S4. This includes solving prefix equations in a more parallel way and using information about failed prefix unifications directly in the classical proof search carried out in the first step. For a restricted set of modal first-order formulae, the embedding and translation nano-M2C into classical logic can also be used for the non-constant domains. This set includes, e.g., formulae for which the proof does not assign Eigenvariables to term variables.

Acknowledgments

The author would like to thank Stephan Schulz for providing the E prover.

[10] Leo-III does not support the QMLTP syntax, but previous evaluations show that it solves fewer problems than MleanCoP and, hence, also fewer problems than nanoCoP-M.

References

[1] M. Fitting, R. L. Mendelsohn, First-order modal logic, Springer, 1998.

[2] 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, volume 5195 of *LNAI*, Springer, 2008, pp. 283–291.

[3] J. Otten, MleanCoP: A connection prover for first-order modal logic, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), IJCAR 2014, volume 8562 of *LNAI*, Springer, 2014, pp. 269–276.

[4] J. Otten, The nanoCoP 2.0 connection provers for classical, intuitionistic and modal logics, in: A. Das, S. Negri (Eds.), TABLEAUX 2021, volume 12842 of *LNAI*, Springer, 2021, pp. 236–249.

[5] L. A. Wallen, Automated Deduction in Nonclassical Logics, MIT Press, Cambridge, 1990.

[6] W. Bibel, Automated Theorem Proving, Artificial intelligence, F. Vieweg und Sohn, 1987.

[7] J. Otten, Restricting backtracking in connection calculi, AI Commun. 23 (2010) 159–182.

[8] J. Otten, Implementing connection calculi for first-order modal logics, in: K. Korovin, S. Schulz, E. Ternovska (Eds.), IWIL 2012, volume 22 of *EPiC Series in Computing*, EasyChair, 2012, pp. 18–32.

[9] G. Gentzen, Untersuchungen über das Logische Schließen, Mathematische Zeitschrift 39 (1935) 176–210, 405–431.

[10] J. Otten, Non-clausal connection calculi for non-classical logics, in: R. Schmidt, C. Nalon (Eds.), TABLEAUX 2017, volume 10501 of *LNAI*, Springer, Cham, 2017, pp. 209–227.

[11] J. Otten, C. Kreitz, T-string unification: Unifying prefixes in non-classical proof methods, in: P. Miglioli, U. Moscato, D. Mundici, M. Ornaghi (Eds.), TABLEAUX '96, volume 1071 of *LNAI*, Springer, 1996, pp. 244–260.

[12] J. Otten, Clausal connection-based theorem proving in intuitionistic first-order logic, in: B. Beckert (Ed.), TABLEAUX 2005, volume 3702 of *LNAI*, Springer, 2005, pp. 245–261.

[13] W. Bibel, Matings in matrices, Commun. ACM 26 (1983) 844–852.

[14] J. Otten, W. Bibel, leanCoP: lean connection-based theorem proving, Journal of Symbolic Computation 36 (2003) 139–161.

[15] J. Otten, nanoCoP: A non-clausal connection prover, in: N. Olivetti, A. Tiwari (Eds.), IJCAR 2016, volume 9706 of *LNAI*, Springer, Heidelberg, 2016, pp. 302–312.

[16] S. Schulz, S. Cruanes, P. Vukmirović, Faster, higher, stronger: E 2.3, in: P. Fontaine (Ed.), CADE 27, volume 11716 of *LNCS*, Springer, 2019, pp. 495–507.

[17] T. Raths, J. Otten, The QMLTP problem library for first-order modal logics, in: B. Gramlich, et al. (Eds.), IJCAR 2012, volume 7364 of *LNAI*, Springer, 2012, pp. 454–461.

[18] H. J. Ohlbach, A resolution calculus for modal logics, in: E. L. Lusk, R. A. Overbeek (Eds.), 9th CADE, volume 310 of *LNCS*, Springer, 1988, pp. 500–516.

[19] H. J. Ohlbach, Optimized translation of multi modal logic into predicate logic, in: A. Voronkov (Ed.), LPAR'93, volume 698 of *LNCS*, Springer, 1993, pp. 253–264.

[20] A. Steen, C. Benzmüller, The higher-order prover Leo-III, in: D. Galmiche, S. Schulz, R. Sebastiani (Eds.), ICAR 2018, volume 10900 of *LNAI*, Springer, 2018, pp. 108–116.