Solving Quantified Modal Logic Problems by Translation to Classical Logics

Alexander Steen Geoff Sutcliffe Christoph Benzmüller

5 February 2025

Abstract

This article describes an evaluation of Automated Theorem Proving (ATP) systems on problems taken from the QMLTP library of first-order modal logic problems. Principally, the problems are translated to both typed first-order and higher-order logic in the TPTP language using an embedding approach, and solved using first-order resp. higher-order logic ATP systems and model finders. Additionally, the results from native modal logic ATP systems are considered, and compared with the results from the embedding approach. The findings are that the embedding process is reliable and successful when state-of-the-art ATP systems are used as backend reasoners, The first-order and higher-order embeddings perform similarly, native modal logic ATP systems have comparable performance to classical systems using the embedding for proving theorems, native modal logic ATP systems are outperformed by the embedding approach for disproving conjectures, and the embedding approach can cope with a wider range of modal logics than the native modal systems considered.

Keywords: Non-classical logics, Quantified modal logics, Higher-order logic, First-order logic, Automated theorem proving

1 Introduction

Automated Theorem Proving (ATP) systems try to prove formally and fully automatically that a conjecture is entailed by a given set of premises. For first-order quantified modal logics [17] there exist a few ATP systems, including GQML [53], MleanTAP, MleanSeP, MleanCoP [31], and the more recent nanoCoP-M 2.0 [32]. Unfortunately, none of them support all 15 modal logics in the modal cube [19]. For example, MleanTAP, MleanCoP, and nanoCoP-M 2.0 support only the **D**, **M** (aka **T**), **S4**, and **S5** logics; GQML supports only **K**, **D**, **K4**, **M**, and **S4**; and MleanSeP supports only **K**, **K4**, **D**, **D4**, **S4**, and **M**. As a consequence, even with the most versatile (in terms of the number of different modal logics supported) ATP systems from this list only 40% of all modal logics from the modal cube are covered. This effect is multiplied when taking

into account further modal logic properties such as different domain semantics [17], rigidity of terms [12], and multi-modal combinations [12] of these properties. For example, non-rigid terms are covered by only GQML, and none of the listed systems support decreasing domains (also referred to as anti-monotonic frames [17]). Further notions like global and local assumptions [17, 11] are unsupported, but are important, e.g., for applications in computational philosophy [9].

An alternative to developing native modal logic ATP systems is to translate modal logic problems to classical first-order logic [30, 14] or higher-order logic [6, 8, 7], and then solve the first- resp. higher-order logic problems. Translations to higher-order logic were primarily motivated by the automation of modal logics with propositional, second-order or generally higher-order quantification; they turned out to provide convincing automation for standard first-order modal logics with acceptable performance and, in particular, unchallenged flexibility [21, 3, 45]. In particular, this indirect approach provides a rich ecosystem for adjusting the different properties of the modal logic under consideration. Earlier results using classical logic ATP systems to solve translated modal logic problems are presented in [6, 8, 21, 45]. Over the last decade there has been substantial progress in the development of ATP systems for classical first-order and higher-order logic, including those used in this work. In the work reported in this paper three state-of-the-art ATP systems – three theorem provers and one model finder – were evaluated on translations of first-order modal logic problems taken from the QMLTP library [37]. The theorem proving ATP systems are E 3.0.03 [38, 56], Leo-III 1.7.8 [42], and Vampire 4.8 [10]. The model finding ATP system is Nitpick 2016 [13].

Extending earlier work [45], the goals of this work were:

1. Test the process of translating first-order modal logic problems into classical first- and higher-order logic using a shallow embedding.
2. Evaluate multiple first- and higher-order ATP systems' abilities to solve the embedded modal logic problems.
3. Compare the performance of native modal logic ATP systems with the embedding approach.

The first goal provides feedback regarding the process, ensuring that the classical first- and higher-order logic problems faithfully reflect the original non-classical modal logic problems. In particular, the soundness and practicality of this translation process is tested by a mutual comparison of prover results; discrepancies in prover results would indicate potential errors. The second goal provides information about which first- and higher-order ATP systems can be most effectively used as a backend in a tool chain that solves modal logic problems by translation to first- resp. higher-order logic. The third goal provides an interesting evaluation of the efficacy of the embedding approach in comparison to reasoning directly in modal logic.

The remainder of this article is structured as follows: Section 2 briefly introduces modal logic, the QMLTP library, and higher-order logic. Section 3

introduces the TPTP languages that have been used in this work for encoding the modal logic problems and their corresponding translations to classical logic. Section 4 describes how non-classical logic problems can be translated to classical first- and higher-order logic problems, with a particular emphasis on modal logic as used in the QMLTP. Section 5 describes the experimental setup, and the results of the evaluation. A performance comparison with native modal logic ATP systems is provided. Section 6 concludes and sketches further work.

2 Preliminaries

2.1 Modal Logic

First-order modal logic (FOML) is a family of logic formalisms extending classical first-order logic (FOL, here without equality) with the unary modal operators $\Box$ and $\Diamond$. FOML terms are defined as for FOL. The modal logic **K** is the smallest logic that includes all FOL tautologies, all instances of the axiom scheme K $(\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi))$, and is closed with respect to modus ponens (from $\varphi \to \psi$ and φ infer ψ) and necessitation (from φ infer $\Box\varphi$). In modal logics considered in this work the modal operators are dual notions, i.e., $\Box\varphi$ is equivalent to $\neg\Diamond\neg\varphi$. Further axiom schemes are assumed for stronger modal logics [17].

K can alternatively be characterized by Kripke structures [27]. For a FOML language over signature Σ, a first-order Kripke frame is a tuple $M = (W, R, \mathcal{D}, \mathcal{I})$, where W is a non-empty set (the possible worlds), $R \subseteq W \times W$ is a binary relation on W (the accessibility relation), $\mathcal{D} = \{D_w\}_{w \in W}$ is a family of non-empty sets D_w (the domain of world w) and $\mathcal{I} = \{I_w\}_{w \in W}$ is a family of interpretation functions I_w, one for each world $w \in W$, that map the symbols of Σ in world $w \in W$ to adequate denotations over D_w. In Kripke semantics the truth of a formula ϕ with respect to M and a world $w \in W$, written $M, w \models \phi$, is defined as usual [17]. For stronger modal logic systems, restrictions may be imposed on the relation R, e.g., modal logic **D** is characterized by the class of Kripke frames where R is serial. Similar correspondence results exist for the other common modal logics [19]. Multi-modal logics generalize the language with multiple (indexed) modalities, where the accessibility relation R is replaced with a set of relations $\mathcal{R} = \{R_i\}_{i \in I}$, one for each indexed modality $\Box_i$, $i \in I$, where I is some index set. Each indexed modality can be assumed to satisfy stronger properties by restricting its accessibility relation R_i.

Following Fitting and Mendelsohn [17], local consequence of a formula φ in FOML is defined with respect to a set of global assumptions G and a set of local assumptions L, written $G \models L \to \varphi$. If G is empty, this reduces to the common notion of local consequence. See [17] for the precise definition.

Further variations of the Kripke semantics are possible; three independent refinements are well-discussed in the literature. Firstly, quantification semantics may be specialized by domain restrictions [17]: In *constant domains* semantics (also called possibilist quantification), all domains are assumed to coincide, i.e.,

$D_w = D_v$ for all worlds $w, v \in W$. In *cumulative domains* semantics, the domains are restricted such that $D_w \subseteq D_v$ whenever $(w, v) \in R$, for all $w, v \in W$. In *decreasing domains* semantics, it holds that $D_v \subseteq D_w$ whenever $(w, v) \in R$, for all $w, v \in W$. In *varying domains* semantics (also called actualist quantification) no restriction is imposed on the domains. These semantic restrictions can be equivalently characterized in a proof-theoretic way using the (converse) Barcan formulae [2, 17]. Secondly, constancy restrictions of the interpretation of constant and function symbols across different possible worlds may be applied. The interpretation is *rigid* if $I_w(c) = I_v(c)$ and $I_w(f) = I_v(f)$ for each constant symbol $c \in \Sigma$ and function symbol $f \in \Sigma$, and all worlds $w, v \in W$. If this is not the case the interpretation is *flexible*. This distinction is also sometimes referred to as rigid vs. flexible (or: world-dependent) *designation*. Details are discussed in [17]. Thirdly, the domain of admissible interpretations of terms may be constrained. So-called *local terms* [37] require that, for every world w, every ground term t denotes at w an object that exists at w, in particular it holds that $\mathcal{I}_w(c) \in \mathcal{D}_w$ and $\mathcal{I}_w(f)(d_1, \ldots, d_n) \in \mathcal{D}_w$ for every world w, every constant symbol $c \in \Sigma$, every function symbol $f \in \Sigma$ of arity n, and objects $d_1, \ldots, d_n \in \mathcal{D}_w$. If such a restriction is not imposed, terms are sometimes referred to as *global*. Local terms should not be confused with local assumptions (local consequence): The former speaks about restrictions on the interpretation of terms, the latter about the consequence relation.

2.2 The QMLTP library

The Quantified Modal Logics Theorem Proving (QMLTP) library[1] provides a problem collection of FOML problems, information about ATP systems for FOML, and performance results from the systems on the problems in the library (last updated around 2012). It is modelled after the TPTP library for classical logic [48]. The problem collection is divided into 11 domains in five groups:

- The `APM` domain of "mixed applications". There are 10 problems.
- The `G??` domains are Gödel encodings of TPTP problems viewed as intuitionistic problems. There are 245 problems.
- The `MML` multi-modal logic problems. There are 20 problems, each with a particular logic specification.
- The `NLP` and `SET` domains of classical logic problems, treated as modal logic problems. They have no modal connectives, but are are included in the evaluation because they are part of the QMLTP. There are 5 `NLP` and 75 `SET` problems.
- The `SYM` domain of syntactic modal logic problems. There are 245 problems.

[1] `http://www.iltp.de/qmltp`

Overall there are 600 problems, of which 580 are mono-modal and 20 are multi-modal. The problems assume rigid constants, local terms, and local consequence (i.e., all axioms in the problem files are local). Each problem is documented with the expected result for all combinations of constant, cumulative, and varying domains, in modal systems **K**, **D**, **M**, **S4**, and **S5**. The problems do not have expected results for decreasing domains, assumedly because of the absence of native modal logic ATP systems that support decreasing domains. In general, each problem might have a different expected result for each combination of logic properties (i.e., being a theorem or not). There are problems that are unprovable for any combination of modal logic properties, problems that are provable in some but not all combinations, and problems that are provable in all combinations of properties considered in the QMLTP. The FOML framework from Section 2.1 subsumes these combinations of properties.

The QMLTP library also defines a lightweight syntax extension of the TPTP's FOF language, denoted `qmf`, for expressing FOML. The `qmf` syntax is not introduced here, and instead the TPTP language for modal logic, introduced in Section 3, is used.

2.3 Higher-Order Logic

There are many quite different frameworks that fall under the general label "higher-order". The notion reaches back to Frege's original predicate calculus [18]. Church introduced simple type theory [15], a higher-order framework built on his simply typed λ-calculus, employing types to reduce expressivity and to remedy paradoxes and inconsistencies. Variants and extensions of Church's simple type theory have been the logic of choice for interactive proof assistants such as HOL4 [22], HOL Light [23], PVS [34], Isabelle/HOL [29], and OMEGA [39]. A variation of simple type theory is extensional type theory [24, 5]. It is a common basis for higher-order ATP systems, including those used in this work. For the remainder of this article "higher-order logic" (HOL) is therefore synonymous with extensional type theory, and is the intended logic of the TPTP THF language [50] used in this work (see Section 3). The semantics is the general semantics (or Henkin semantics), due to Henkin and Andrews [24, 1, 4]. See [5, 4] for a full introduction to higher-order logic syntax and semantics.

3 The TPTP Languages

The TPTP languages [49] are human-readable, machine-parsable, flexible and extensible languages, suitable for writing both ATP problems and ATP solutions[2]. In this section the general structure of the TPTP languages is reviewed, and the two specific TPTP languages used for non-classical logics are presented. The full syntax of the TPTP languages is available in extended BNF form.[3]

[2]The development of TPTP World standards for writing ATP solutions beyond common derivations and models is still necessary; see, e.g., [33]

[3]`http://www.tptp.org/TPTP/SyntaxBNF.html`

The top-level building blocks of the TPTP languages are *annotated formulae*, in the form:

language`(`*name*`,` *role*`,` *formula*`,` *source*`,` *useful_info*`).`

The *language*s supported are clause normal form (`cnf`), first-order form (`fof`), typed first-order form (`tff`), and typed higher-order form (`thf`). `tff` and `thf` are each used for multiple languages in the TPTP language hierarchy, described below. The *name* assigns a (unique) identifier to each formula, for referring to it. The *role*, e.g., `axiom`, `lemma`, `conjecture`, defines the use of the formula in an ATP system. In the *formula*, terms and atoms follow Prolog conventions. The TPTP language also supports interpreted symbols, including: "the type of types" `$tType`; types for individuals `$i` (ι) and booleans `$o` (o); types for numbers `$int` (integers), `$rat` (rationals), and `$real` (reals); numeric constants such as 27, 43/92, -99.66; arithmetic predicates and functions such as `$greater` and `$sum`; the truth constants `$true` and `$false`. The basic logical connectives are `^`, `!`, `?`, `@`, `~`, `|`, `&`, `=>`, `<=`, `<=>`, and `<~>`, for λ, $\forall$, $\exists$, higher-order application, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$, and $\oplus$ respectively. Equality and inequality are expressed as the infix operators `=` and `!=`. The *source* and *useful_info* are optional extra-logical information about the origin and useful details about the formula. See [48] or the TPTP web site `https://www.tptp.org` for all the details. An example of an annotated first-order formula defining the set-theoretic union operation, supplied from a file named `SET006+1.ax`, is ...

```
fof(union,axiom,
    ( ! [X,A,B] :
        ( member(X,union(A,B))
      <=> ( member(X,A) | member(X,B) ) ),
    file('SET006+0.ax',union),
    [description('Definition of union'), relevance(0.9)]).
```

The TPTP has a hierarchy of languages that ends at the non-classical languages used in this work. The languages are:

- Clause normal form (CNF), which is the "assembly language" of many modern ATP systems.
- First-order form (FOF), which hardly needs introduction.
- Typed first-order form (TFF), which adds types and type signatures, with monomorphic (TFF) and polymorphic (TF1) variants.
- Typed extended first-order form (TXF), which adds Boolean terms, Boolean variables as formulae, tuples, conditional expressions, and let expressions. TXF has monomorphic (TX0) and polymorphic (TX1) variants.
- Typed higher-order form (THF), which adds higher-order notions including curried type declarations, lambda terms, partial application, and connectives as terms. THF has monomorphic (THF) and polymorphic (TH1) variants. THF is the TPTP language used for HOL.
- Non-classical forms (NXF and NHF), which add *non-classical connectives* and *logic specifications*. The non-classical typed extended first-order

form (NXF) builds on TXF, and the non-classical typed higher-order form (NHF) builds on THF. NXF and NHF are the TPTP languages used for FOML.

3.1 Non-Classical Connectives

The non-classical connectives of NXF and NHF have the form `{$`*connective_name*`}`. A connective may optionally be parameterized to reflect more complex non-classical connectives, e.g., in multi-modal logics where the modal operators are indexed, or in epistemic logics [55] where the common knowledge operator can specify the agents under consideration. The form is `{$`*connective_name*`(`$param_1$`,...,`$param_n$`)}`. If the connective is indexed the index is given as the first parameter prefixed with a `#`. All other parameters are key-value assignments. In NXF the non-classical connectives are applied in a mixed "higher-order applied"/"first-order functional" style, with the connectives applied to a `()`ed list of arguments.[4] In NHF the non-classical connectives are applied in usual higher-order style, with curried function applications using the application operator `@`. For FOML the connectives are `{$box}` and `{$dia}` for $\Box$ and $\Diamond$. In the context of multi-modal FOML they are `{$box(#i)}` and `{$dia(#i)}` for $\Box_i$ and $\Diamond_i$, respectively, where `#i` is a representation of an index i. Figure 1 illustrates the use of connectives from some (not further specified), modal, alethic modal, and epistemic logics.

Figure 1: Non-classical connective examples

```
tff(pigs_fly_decl,type, pigs_fly: $o ).

tff(flying_pigs_impossible,axiom,
    ~ {$possible} @ (pigs_fly) ).

tff(alice_knows_pigs_dont_fly,axiom,
    {$knows(#alice)} @ (~ pigs_fly) ).

tff(something_is_necessary,axiom,
    ? [P: $o] : {$necessary} @ (P) ).


tff(a_decl,type, a: $i ).
tff(f_decl,type, f: $i > $o ).

tff(reasonable,conjecture,
    ( ! [X: $i] : ( {$box} @ (f(X)) )
   => ( {$box} @ (! [X: $i] : f(X)) ) ) ).

tff(silly,conjecture,
    ( {$box(#a)}
    @ ( {$dia(#a)} @ (! [X: $i] : f(X)) ) ) ).
```

```
thf(positive_decl,type,
    positive: ($i > $o) > $o ).

thf(self_identity_is_positive,axiom,
    {$necessary} @
      ( positive @ ^ [X:$i] : (X = X) ) ).

thf(alice_and_bob_know,axiom,
    {$common($agents:=[alice,bob])} @
      ( positive @ ^ [X:$i] : (X = X) ) ).

thf(everything_is_possibly_positive,axiom,
    ! [P: $i > $o] :
      ( {$possible} @ (positive @ P) ) ).
```

[4] This slightly unusual form was chosen to reflect the first-order functional style, but by making the application explicit the formulae can be parsed in Prolog – a long standing principle of the TPTP languages [52].

3.2 Logic Specifications

In contrast to classical logic, in the world of non-classical logics the intended logic cannot be inferred from the language used for the formulae – the same syntactical language can host different logics with different proof-theoretic inferences on the formulae. For example, when reasoning about metaphysical necessity, modal logic **S5** is usually used, but when reasoning about deontic necessities a more suitable choice might be modal logic **D**. Nevertheless, modal logics **S5** and **D** share the same syntactical language. It is therefore necessary to provide (meta-)information that specifies the logic to be used.

A new kind of TPTP annotated formula is used to specify the logic and its properties, The annotated formulae has the role `logic`, and has a "logic specification" as its formula. A logic specification consists of a defined logic (family) name identified with a list of properties, e.g., in NXF ...

`tff(`*name*`,logic,`*logic_name* `==` *properties*`).`

where *properties* is a `[]` bracketed list of key-value identities ...

property_name `==` *property_value*

Each *property_name* is a TPTP defined or system symbol. Each *property_value* is either a term (often a defined constant) or a `[]` bracketed list that optionally starts with a term (often a defined constant), and otherwise contains key-value identities. If the first element of a *property_value* list is a term then that is the default value for all cases that are not specified by the following key-value identities.

In this work logic specifications for FOML are used. The TPTP defines the constant `$modal` as the *logic_name* for modal logics. The `$modal` family follows the generalized notion of consequence for modal logics by Fitting and Mendelsohn [17], as briefly introduced in Section 2.1, which allows for both local and global premises in problems [11, Ch 1.5]. Annotated formulae with the `axiom` role are global premises (true in all worlds), and those with the `hypothesis` role are local premises (true in the current world). These default role readings can be overridden using *subroles*, e.g., a formula with the role `axiom-local` is local, and a formula with the role `hypothesis-global` is global. The following properties can be specified for `$modal` logics (as discussed in Section 2.1):

- The `$domains` property specifies restrictions on the domains across the accessibility relation. The possible values are `$constant`, `$varying`, `$cumulative`, and `$decreasing`.
- The `$designation` property specifies whether symbols are interpreted as `$rigid`, i.e., interpreted as the same domain element in every world, or as `$flexible`, i.e., possibly interpreted as different domain elements in different worlds.
- The `$terms` property specifies whether term restrictions are `$local` or `$global`.
- The `$modalities` property specifies the modality of connectives, either all connectives, or by index. Possible values are defined for well-known modal logic systems, e.g., `$modal_system_K`, and individual modal axiom

schemes, e.g., `$modal_axiom_5`. They refer to the corresponding systems and axioms of the modal logic cube [19].

In order to reduce misunderstandings there are no default values. It is an error if a problem does not specify all relevant properties (but irrelevant ones may be omitted, e.g., `$domains` may be omitted if the problem does not contain any quantification).

A simple example from modal logic **S5** with constant domains, local terms, and with rigid constants, is ...

```
tff(simple_spec,logic,
    $modal == [
      $domains == $constant,
      $designation == $rigid,
      $terms == $local,
      $modalities == $modal_system_S5 ] ).
```

Multi-modal logics are specified by enumerating the modalities of the connectives. Here is an example of a logic specification with constant domains, rigid symbols, and local terms. Modalities are generally **K**, but $\Box_a$ satisfies modal axiom K and B, and $\Box_b$ is a **S4** modality.

```
thf(multi_spec,logic,
    $modal == [
      $domains == $constant,
      $designation == $rigid,
      $terms == $local
      $modalities == [ $modal_system_K,
        {$box(#a)} == [ $modal_axiom_K, $modal_axiom_B ],
        {$box(#b)} == $modal_system_S4 ] ] ).
```

Detailed information on the general non-classical TPTP format and its associated TPTP infrastructure is presented in [43].

4 Shallow Embeddings into Classical Logic

Shallow embeddings of a source logic into a target logic are translations that encode the semantics of formulae of the source logic as terms or formulae of the target logic. This is in contrast to so-called deep embeddings that encode the source logic formulae as uninterpreted data (usually an inductively defined datatype), and define meta-theoretical notions such as interpretation and satisfiability as functions and predicates. An in-depth discussion of pros and cons of both approaches is presented in [20]. The shallow embedding of modal logics into classical higher-order logic used in this work is based on earlier work [6, 7, 8, 21]. The shallow embedding of modal logics into classical (typed) first-order logic is commonly referred to as *standard translation* [30, 14] and widely known. The main ideas of the embedding into higher-order logic are informally sketched in

the following. The embedding into first-order logic is conceptually very similar and based on the standard translation of [14], which is not explained in detail here. The first-order embedding was suitably augmented to cover all the different modal logic parameters, as explained below.

The notion of Kripke semantics [11, 12] is simulated in HOL as follows: A new type μ is introduced as the type of possible worlds, and a binary relation (predicate) symbol $r^i_{\mu\to\mu\to o}$, where o is the classical type of truth values in HOL, models the (indexed) accessibility relation R^i between worlds. Since the truth of a formula φ in FOML is established through assessing the truth of φ with respect to all worlds, formulae of FOML are identified with HOL predicates of type $\mu \to o$, abbreviated σ in the following. Connectives are defined accordingly, evaluating a composite formula with respect to a given world. For example, the box operator, negation, disjunction, and constant-domain universal quantification over objects of type τ are defined as follows (subscripts denote types: a type $\tau \to \nu$ is the type of total functions from objects of type τ to objects of type ν; expressions of form $\lambda X_\tau.T_\nu$ are anonymous functions mapping their parameters X_τ to some object of type ν as given by term T):

$$\begin{array}{ll}
\Box^i_{\sigma\to\sigma} & := \lambda S_\sigma.\lambda W_\mu.\forall V_\mu.\ \neg(r^i\ W\ V) \vee S\ V \\
\neg_{\sigma\to\sigma} & := \lambda S_\sigma.\lambda W_\mu.\ \neg(S\ W) \\
\vee_{\sigma\to\sigma\to\sigma} & := \lambda S_\sigma.\lambda T_\sigma.\lambda W_\mu.\ (S\ W) \vee (T\ W) \\
\Rightarrow_{\sigma\to\sigma\to\sigma} & := \lambda S_\sigma.\lambda T_\sigma.\lambda W_\mu.\ (S\ W) \Rightarrow (T\ W) \\
\Pi^\tau_{(\tau\to\sigma)\to\sigma} & := \lambda P_{\tau\to\sigma}.\lambda W_\mu.\forall X_\tau.\ P\ X\ W
\end{array}$$

A HOL term that simulates the original FOML formula is then obtained by a translation function $\lceil . \rceil$ that recursively translates all occurrences of FOML connectives with their embedded counterparts. The truth of φ at world w in FOML is reduced to truth of the resulting HOL term applied to an object of type μ that represents w. Finally, the meta-logical notion of consequence is similarly mapped to HOL predicates [7].

As an example, the step-wise translation of the FOML formula $\big(\forall x\,\Box P(x) \Rightarrow \Box\forall x\,P(X)\big)$ via $\lceil . \rceil$ into HOL is as follows (assuming P is an unary predicate symbol):

$$\begin{array}{ll}
 & \lceil \forall x\,\Box P(x) \Rightarrow \Box\forall x\,P(x) \rceil \\
= & \lambda W_\mu.\ \big(\lceil \forall x\,\Box P(x) \rceil\ W\big) \Rightarrow \big(\lceil \Box\forall x\,P(x) \rceil\ W\big) \\
= & \lambda W_\mu.\ \big((\lambda W_\mu.\forall x_\iota.\ \lceil \Box P(x) \rceil\ W)\ W\big) \\
 & \qquad \Rightarrow \big((\lambda W_\mu.\forall V_\mu.\ \neg(r\ W\ V) \vee \lceil \forall x\,P(x) \rceil\ V)\ W\big) \\
= & \lambda W_\mu.\ \Big(\big(\lambda W_\mu.\forall x_\iota.\ (\lambda W_\mu.\forall V_\mu.\ \neg(r\ W\ V) \vee \lceil P(x) \rceil\ V)\ W\big)\ W\Big) \\
 & \qquad \Rightarrow \Big(\big(\lambda W_\mu.\forall V_\mu.\ \neg(r\ W\ V) \vee (\lambda W_\mu.\forall x_\iota.\ \lceil P(x) \rceil\ W)\ V\big)\ W\Big) \\
= & \lambda W_\mu.\ \Big(\Big(\big(\lambda W_\mu.\forall x_\iota.\ \big(\lambda W_\mu.\forall V_\mu.\ \neg(r\ W\ V) \vee (\lambda W_\mu.\ P_{\iota\to\sigma}\ x\ W)\ V\big)\ W\big)\ W\Big) \\
 & \qquad \Rightarrow \Big(\Big(\lambda W_\mu.\forall V_\mu.\ \neg(r\ W\ V) \vee \big(\lambda W_\mu.\forall x_\iota.\ (\lambda W_\mu.\ P_{\iota\to\sigma}\ x\ W)\ W\big)\ V\Big)\ W\Big)
\end{array}$$

After the FOML formula structure has been translated into the HOL term above, it can be simplified by β-normalization to:

$$\lambda W_\mu.\, \big(\forall x_\iota.\, \forall V_\mu.\, \neg(r\ W\ V) \vee (P_{\iota\to\sigma}\ x\ V)\big) \\ \Rightarrow \big(\forall V_\mu.\, \neg(r\ W\ V) \vee (\forall x_\iota.\, P_{\iota\to\sigma}\ x\ V)\big)$$

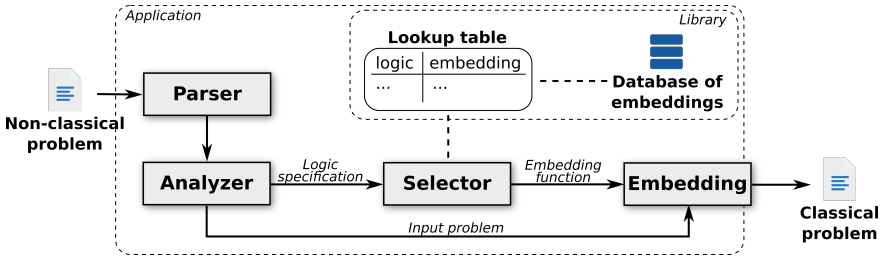
Similarly, the notions of global and local consequence are defined as HOL terms, both of which finally ground the embedded modal logic formulas (terms of type σ) to HOL formulas (terms of type o). The first-order based translation works analogously, only that the convenient λ-functions of HOL (that act as local binding mechanism for the world parameter) are replaced by explicit parameters in the recursive embedding function so that the binding of worlds is handled on the meta-level. The first-order and higher-order embedding of the above example formula is displayed in full detail in Appendix A.

The strength of shallow embeddings comes not only from the reuse of existing classical reasoning technology, but also from the ability to provide a uniform approach for automating many different non-classical logics [3], including the modal logics used in this work. All modal logic systems in the modal logic cube can be embedded by including additional formulae in the embedding process that restrict the encoded accessibility relation r^i according to the corresponding properties of R^i in the given modal logic. For example, modal logic **M** additionally assumes the axiom scheme $\Box_i\varphi \to \varphi$ that corresponds to reflexive frames, and the analogous HOL formula $\forall W_\mu.\ r^i\ W\ W$ is included in the embedding process. For varying domains the definition of universal quantification is adjusted by *relativization*, i.e. to include an extra *exists-in-world* predicate eiw^τ that mimics the different domains [6]. Cumulative and decreasing domains are then easily characterized by imposing additional constraints on eiw^τ [21].

Following the promising results of earlier work [45, 9], the embedding process automatically uses an optimization for embeddings of **S5** logics in the mono-modal context – the alternative characterization of **S5** employing frames with a universal accessibility relation, as opposed to requiring an equivalence relation, is used. The two variants are known to be equivalent with respect to provability, and the universal variant is more efficient for proof automation. Similarly, it can also easily be shown that in mono-modal **S5**, cumulative, constant, and decreasing domains coincide with respect to provability (**S5** frames are equivalence relations, in particular they are symmetric; so every monotone frame is also anti-monotone, and thus constant) [36, 25]. Using this optimization, the embedding process can use the more light-weight constant domain encoding of **S5** problems with cumulative or decreasing domains, thus avoiding complicating the output with relativized quantifiers.

The Leo-III system implements the entire embedding process, from reading non-classical logic problems in the NXF and NHF languages, embedding the problems into HOL using the THF language, and using higher-order reasoning to (attempt to) solve the higher-order problems. This is realized using the Logic Embedding Tool [40, 41] that implements the embedding process, as shown in Figure 2: The Logic Embedding Tool will read and parse the input problem

Figure 2: Embedding process (picture taken from [40])



formulated in NTF or NHF, extract its logic specification, choose the right logic embedding function based on the logic specification and its parameters from a database of available logic embeddings, and finally return the embeddings functions results in classical TFF or THF. As experimental feature, the tool can also output polymorphically typed result problems in the TF1 and TH1 languages, but this has not been thoroughly tested and evaluated yet and is thus not in the scope of this work. While Leo-III includes the tranlation code as an internal library, it is also available as a stand-alone tool that can be used as pre-processor to any TPTP-compliant higher-order ATP system. It is thus possible to replace Leo-III as the backend reasoning engine, as evaluated in Section 5.2. The tool is open-source and available via Zenodo [41] and GitHub[5].

5 Evaluation

5.1 Experimental setup

The QMLTP library v1.1[6] was used for the evaluation. During the preparation of the problems it was noticed that some original QMLTP problems have syntax errors (missing parentheses, etc.), and some problems use equality without an axiomatization (including use of the infix = predicate that is interpreted in many ATP systems). These issues were corrected for the evaluation.[7]

The sequence of steps taken to prepare the mono-modal problems was:

- Convert the mono-modal problems from the QMLTP syntax to the NXF language. In particular, the QMLTP `#box` and `#dia` connectives were converted to `{$box}` and `{$dia}`, and then applied to the formula argument. The axioms' roles were set to `axiom-local`, to reflect the QMLTP's local consequence.

[5] `https://github.com/leoprover/logic-embedding`

[6] `http://www.iltp.de/qmltp/problems.html`

[7] The conditions stated in [37] for "presenting results of modal ATP systems based on the QMLTP library" say that "no part of the problems may be modified". As such the results presented in this article for the corrected versions of the QMLTP problems cannot be called "results for problems from the QMLTP". But pragmatically, the results on the set of (corrected) problems are comparable with the results on the original QMLTP problems.

- Add each of the 15 logic specifications (three domain types combined with five modal systems) to each of the non-MML NXF problems to produce a total of 8700 mono-modal NXF problems.
- Translate the 8700 NXF problems to both classical TFF and classical THF using the logic embedding tool.

In the MML domain, there is just one logic specification for each of the 20 problems because the problems come with a single intended semantics. The pre-processing steps were the same as for the mono-modal problems, producing 20 multi-modal NXF problems, and hence 20 embedded TFF problems and 20 embedded THF problems. In total the evaluation set contains 8720 NXF problems for the native modal logic systems, and their corresponding embeddings embeddings in TFF and THF.

E 3.0.03, Leo-III 1.7.8, Vampire 4.8 were used to prove TFF and THF conjectures, and to disprove TFF conjectures – none of them are able to disprove THF conjectures. Nitpick 2016 was used to disprove THF conjectures – that's all it can do. (An analysis of further higher-order backend systems is presented in earlier work [45].) For the native modal logic ATP systems, MleanCoP 1.3, which implements modal connection calculi, and nanoCoP-M 2.0, which implements a non-clausal connection calculus, were used to prove and disprove NXF conjectures. These two systems are the strongest native modal logic ATP systems to date [32, 31, 6].

During the evaluation it was observed that MleanCoP and nanoCoP-M give conflicting results (i.e., one claiming the conjecture to be a theorem, the other claiming it to be a non-theorem) for 11 problems from the GSY and SYM domains. After manual inspection and cross-validation using results from the embedded variants, this suggested unsoundness in nanoCoP-M. As the authors were not successful in contacting the developer of both systems, the need for a further investigation of this issue currently remains. For the purpose of this evaluation the 11 conflicting results from nanoCoP-M were removed, but the unsoundness renders nanoCoP-M's results an optimistic performance estimation. The authors of this paper believe that the extent of the unsoundness is quite limited, and thus the system is still included in the evaluation.

MleanCoP and nanoCoP-M are implemented in Prolog; ECLiPSe Prolog version 5.10 is used. All the ATP systems except Nitpick were run on the StarExec [46] Miami cluster with a 60s wall clock and 480 CPU time limit. The StarExec Miami computers have an octa-core Intel Xeon E5-2667 3.20 GHz CPU, 128 GiB memory, and run the CentOS Linux release 7.4.1708 operating system. Nitpick was run on a separate computer with a 60s wall clock time limit[8]. The computer has an octa-core Intel Xeon E5-2609 2.50 GHz CPU, 64 GiB memory, and the CentOS Linux release 7.9.2009 operating system. All source problems and the evaluation results are available as supplemental material to this article via Zenodo [44]. The auxiliary scripts for generating the

[8]Nitpick could not be installed in StarExec. None of the resulting insights are affected because enough resources were allocated for the system to find its solutions – see Sect. 3.1 of [51] for details on *Peter Principle Points* of automated reasoning systems.

problem files are available from the TPTP's non-classical logic GitHub repository[9] (recall from Section 4 that the embedding tool is available separately).

Tables 1, 2, and 3 give the numbers of problems solved. The tables present the numbers of proofs (THM) and disproofs (CSA) for the various logics, the various logic parameters, the two embeddings (THF and TFF) and their union (T?F), the NXF problems used by the native modal logic systems, and various unions ($\cup$) of problems solved. The columns marked with a backslash give the number of problems solved by the indicated column(s) and not solved by column(s) indicated after the backslash, e.g., in T?0 the \NXF column is the number of problems solved by the union of the THF and TFF columns, and not solved by either MleanCoP or nanoCoP-M in the NXF column. The individual ATP systems are abbreviated as "Leo" for Leo-III, "Vam" for Vampire, "Nit" for Nitpick, "MlC" for "MleanCoP", and "nnC" for nanoCoP-M. In each row the best individual system performance is **bolded**.

5.2 Results for Mono-Modal Problems

Table 1, shows the results for modal logics **M**, **D**, **S4** and **S5**, i.e., the logics supported by the native modal logic systems. Table 1 allows for the following observations:

- In eight out of the 12 logic configurations MleanCoP proves more theorems than any other system. In three configurations Vampire proves the most (using the TFF embedding), and in one configuration E proves the most (using the THF embedding).
- MleanCoP generally performs better than nanoCoP-M for proving theorems, Vampire is the strongest system for proving theorems in TFF, and in THF Leo-III is stronger than the other systems in six configurations. Vampire and E are strongest in three configurations each. It is notable that Vampire usually performs better with constant domains.
- On average, in **M** MleanCoP 2.1% more theorems than the best system using an embedding. In **D** this increases to 3.1%, and in **S4** it increases to 6.1%.
- In **S5** Vampire proves the most theorems (using the TFF embedding), which is 9.3% more than MleanCoP and 26.0% more than nanoCoP-M. In general, in **S5**/const and **S5**/cumul all the embedding-based systems are stronger than the native modal logic systems.
- The union of the THF and TFF embedding variants (T?0 in the table) proves only marginally more theorems than either one of them.
- Each prover capable of disproving conjectures using either embedding approach is stronger than both the best native modal prover and the union of the native modal logic systems in every logic configuration.

[9] See the QMLTP directory at `https://github.com/TPTPWorld/NonClassicalLogic`.

- nanoCoP-M generally performs better than MleanCoP for disproving conjectures, in TFF Vampire is always the best system, and in THF Nitpick outperforms every other system in every logic configuration.

- On average, in **M** Nitpick disproves 61.8% more conjectures than nanoCoP-M. In **D** this drops to 30.2%, but in **S4** it increases to 64.2%, and in **S5** it increases to 78.5%.

- The union of the embedding variants often disproves significantly more conjectures than Nitpick alone.

- Both the native modal logic systems and the embedding-based approaches find unique solutions, but only in the embedding approach this is the case for all logic configurations.

- Comparing the embedding variants' unions shows that THF yields the strongest results for proving theorems in 10 out of 12 logic configurations. For disproving conjectures, the THF variant outperforms the TFF variant in every logic configuration.

The presentation of results is continued in Table 2 for **K**, and with summaries over the different logics. Neither MleanCoP nor nanoCoP-M support reasoning in **K**. The results for the THF and TFF systems are included here as their performance can be approximately compared to the reported results [6] for MleanSeP 1.2 – the only other native ATP system for FOML that supports **K** and is indexed in the QMLTP. Previous evaluations indicate that the embedding approach outperforms MleanSeP [6, 45].[10] Since MleanSeP is outperformed in all other logics by MleanCoP and nanoCoP-M [32], it is not further considered in this evaluation.

The main observations from Table 2 are:

- In **K** the TFF embedding proves more theorems than the THF embedding, in every logic configuration. Nitpick outperforms all TFF-based systems for disproving conjectures.

- When summarizing over all logics (rows starting with "∪" in the table), the THF embedding proves slightly more theorems in varying and cumulative domain configurations, while the TFF embedding proves slightly more theorems in constant domain configurations.

- For disproving conjectures the union of the THF-based systems disprove 24.3% more conjectures than the union of the TFF-based systems.

- When summarizing over all logics supported by the native modal logic systems (rows starting with "∪**K**" in the table), the union of THF-based systems proves 1.1% more theorems and disproves 45.0% more conjectures

[10] The evaluation of MleanSeP in [6] uses a higher CPU time limit of 600s, compared to the 60s used in this work, and the hardware employed in is different. Approximate comparisons are still possible.

Table 1: Mono-modal Results

Language	SZS	THF						TFF					T?0		NXF				∪
System		E	Leo	Vam	Nit	∪	\TFF	E	Leo	Vam	∪	\THF	∪	\NXF	MlC	nnC	∪	\T?0	∪
M const	THM	249	251	258	0	262	2	246	251	256	264	4	266	9	**264**	236	270	13	279
	CSA	0	0	0	**181**	181	22	149	0	165	165	6	187	71	110	117	117	1	188
	∪	249	251	258	181	443	24	395	251	**421**	429	10	453	80	374	353	387	14	467
M vary	THM	199	207	203	0	225	8	193	200	215	221	4	229	12	**217**	192	223	6	235
	CSA	0	0	0	**243**	243	49	178	0	200	200	6	249	93	146	149	156	0	249
	∪	199	207	203	243	468	57	371	200	**415**	421	10	478	105	363	341	379	6	484
M cuml	THM	213	230	228	0	248	8	208	220	239	246	6	254	9	**246**	217	251	6	260
	CSA	0	0	0	**202**	202	33	153	0	174	174	5	207	76	124	131	132	1	208
	∪	213	230	228	202	450	41	361	220	**413**	420	11	461	85	370	348	383	7	468
M ∪	THM	661	688	689	0	735	18	647	671	710	731	14	749	30	**727**	645	744	25	774
	CSA	0	0	0	**626**	626	104	480	0	539	539	17	643	240	380	397	405	2	645
	∪	661	688	689	626	1361	122	1127	671	**1249**	1270	31	1392	270	1107	1042	1149	27	1419
D const	THM	195	199	200	0	204	1	193	199	**205**	208	5	209	12	203	171	207	10	219
	CSA	0	0	0	**302**	302	65	199	0	243	243	6	308	102	211	228	228	22	330
	∪	195	199	200	302	506	66	392	199	**448**	451	11	517	114	414	399	435	32	549
D vary	THM	147	158	148	0	169	8	146	151	158	165	4	173	9	**168**	134	170	6	179
	CSA	0	0	0	**340**	340	90	231	0	253	255	5	345	90	257	265	277	22	367
	∪	147	158	148	340	509	98	377	151	411	420	9	518	99	**425**	399	447	28	546
D cuml	THM	162	178	165	0	188	9	162	168	177	185	6	194	10	**187**	155	190	6	200
	CSA	0	0	0	**319**	319	127	170	0	180	197	5	324	96	233	245	251	23	347
	∪	162	178	165	319	507	136	332	168	357	382	11	518	106	**420**	400	441	29	547
D ∪	THM	504	535	513	0	561	18	501	518	540	558	15	576	31	**558**	460	567	22	598
	CSA	0	0	0	**961**	961	282	600	0	676	695	16	977	288	701	738	756	67	1044
	∪	504	535	513	961	1522	300	1101	518	1216	1253	31	1553	319	**1259**	1198	1323	89	1642
S4 const	THM	290	301	319	0	324	12	288	299	306	316	4	328	11	**331**	290	342	25	353
	CSA	0	0	0	**133**	133	26	91	0	112	112	5	138	55	79	83	83	0	138
	∪	290	301	319	133	457	38	379	299	**418**	428	9	466	66	410	373	425	25	491
S4 vary	THM	231	247	234	0	264	8	230	239	251	259	3	267	13	**261**	225	266	12	279
	CSA	0	0	0	**211**	211	64	119	0	152	152	5	216	90	118	120	126	0	216
	∪	231	247	234	211	475	72	349	239	**403**	411	8	483	103	379	345	392	12	495
S4 cuml	THM	255	278	269	0	297	9	252	269	286	295	7	304	11	**316**	278	327	34	338
	CSA	0	0	0	**147**	147	28	101	0	124	124	5	152	55	92	96	97	0	152
	∪	255	278	269	147	444	37	353	269	**410**	419	12	456	66	408	374	424	34	490
S4 ∪	THM	776	826	822	0	885	29	770	807	843	870	14	899	35	**908**	793	935	71	970
	CSA	0	0	0	**491**	491	118	311	0	388	388	15	506	200	289	299	306	0	506
	∪	776	826	822	491	1376	147	1081	807	**1231**	1258	29	1405	235	1197	1092	1241	71	1476
S5 const	THM	444	434	420	0	455	9	442	433	**446**	454	8	463	47	407	355	417	1	464
	CSA	0	0	0	**78**	78	6	76	0	77	77	5	83	41	38	42	42	0	83
	∪	444	434	420	78	533	15	518	433	**523**	531	13	546	88	445	397	459	1	547
S5 vary	THM	343	343	326	0	361	6	340	335	**360**	366	11	372	34	331	284	339	1	373
	CSA	0	0	0	**151**	151	22	124	0	133	134	5	156	62	87	88	94	0	156
	∪	343	343	326	151	512	28	464	335	**493**	500	16	528	96	418	372	433	1	529
S5 cuml	THM	**449**	436	421	0	458	9	442	427	446	456	7	465	48	407	355	417	0	465
	CSA	0	0	0	**78**	78	6	76	0	77	77	5	83	41	38	42	42	0	83
	∪	449	436	421	78	536	15	518	427	**523**	533	12	548	89	445	397	459	0	548
S5 ∪	THM	1236	1213	1167	0	1274	24	1224	1195	**1252**	1276	26	1300	129	1145	994	1173	2	1302
	CSA	0	0	0	**307**	307	34	276	0	287	288	15	322	144	163	172	178	0	322
	∪	1236	1213	1167	307	1581	58	1500	1195	**1539**	1564	41	1622	273	1308	1166	1351	2	1624

than the union of native modal logic systems. The union of TFF-based systems proves 0.5% more theorems and disproves 16.1% more.

- The number of problems solved (proved or disproved) by the union of the THF-based systems over all logics except **K** is 9.3% more than the number solved by the union of the TFF-based systems, and 15.3% more than the number solved by the union of the native modal logic systems.

The evaluation results show that the native modal logic systems prove the most theorems in two thirds of the individual logics. Their performance lead is small (at most 6.1% in modal logics **S4**). The embedding approach performs

Table 2: Mono-modal Results, continued

Language		SZS	THF						TFF					T?0		NXF				∪
System			E	Leo	Vam	Nit	∪	\TFF	E	Leo	Vam	∪	\THF	∪ \NXF		MlC	nnC	∪	\T?0	∪
K	const	THM	180	186	184	0	190	1	183	184	**196**	202	13	203	-	-	-	-	-	-
		CSA	0	0	0	**316**	316	38	262	0	296	298	20	336	-	-	-	-	-	-
		∪	180	186	184	316	506	39	445	184	**492**	500	33	539	-	-	-	-	-	-
K	vary	THM	134	150	140	0	159	8	137	141	**158**	165	14	173	-	-	-	-	-	-
		CSA	0	0	0	**350**	350	86	249	0	267	269	5	355	-	-	-	-	-	-
		∪	134	150	140	350	509	94	386	141	**425**	434	19	528	-	-	-	-	-	-
K	cuml	THM	150	167	155	0	176	7	151	157	**176**	184	15	191	-	-	-	-	-	-
		CSA	0	0	0	**330**	330	93	220	0	238	242	5	335	-	-	-	-	-	-
		∪	150	167	155	330	506	100	371	157	**414**	426	20	526	-	-	-	-	-	-
K	∪	THM	464	503	479	0	525	16	471	482	**530**	551	42	567	-	-	-	-	-	-
		CSA	0	0	0	**996**	996	217	731	0	801	809	30	1026	-	-	-	-	-	-
		∪	464	503	479	996	1521	233	1202	482	**1331**	1360	72	1593	-	-	-	-	-	-
∪	const	THM	1358	1371	1381	0	1435	25	1352	1366	**1409**	1444	34	1469	-	-	-	-	-	-
		CSA	0	0	0	**1010**	1010	157	777	0	893	895	42	1052	-	-	-	-	-	-
		∪	1358	1371	1381	1010	2445	182	2129	1366	**2302**	2339	76	2521	-	-	-	-	-	-
∪	vary	THM	1054	1105	1051	0	1178	38	1046	1066	**1142**	1176	36	1214	-	-	-	-	-	-
		CSA	0	0	0	**1295**	1295	311	901	0	1005	1010	26	1321	-	-	-	-	-	-
		∪	1054	1105	1051	1295	2473	349	1947	1066	**2147**	2186	62	2535	-	-	-	-	-	-
∪	cuml	THM	1229	1289	1238	0	1367	42	1215	1241	**1324**	1366	41	1408	-	-	-	-	-	-
		CSA	0	0	0	**1076**	1076	287	720	0	793	814	25	1101	-	-	-	-	-	-
		∪	1229	1289	1238	1076	2443	329	1935	1241	**2117**	2180	66	2509	-	-	-	-	-	-
∪	∪	THM	3641	3765	3670	0	3980	105	3613	3673	**3875**	3986	111	4091	-	-	-	-	-	-
		CSA	0	0	0	**3381**	3381	755	2398	0	2691	2719	93	3474	-	-	-	-	-	-
		∪	3641	3765	3670	3381	7361	860	6011	3673	**6566**	6705	204	7565	-	-	-	-	-	-
∪\κ	const	THM	1178	1185	1197	0	1245	24	1169	1182	**1213**	1242	21	1266	79	1205	1052	1236	49	1315
		CSA	0	0	0	**694**	694	119	515	0	597	597	22	716	269	438	470	470	23	739
		∪	1178	1185	1197	694	1939	143	1684	1182	**1810**	1839	43	1982	348	1643	1522	1706	72	2054
∪\κ	vary	THM	920	955	911	0	1019	30	909	925	**984**	1011	22	1041	68	977	835	998	25	1066
		CSA	0	0	0	**945**	945	225	652	0	738	741	21	966	335	608	622	653	22	988
		∪	920	955	911	945	1964	255	1561	925	**1722**	1752	43	2007	403	1585	1457	1651	47	2054
∪\κ	cuml	THM	1079	1122	1083	0	1191	35	1064	1084	1148	1182	26	1217	78	**1156**	1005	1185	46	1263
		CSA	0	0	0	**746**	746	194	500	0	555	572	20	766	268	487	514	522	24	790
		∪	1079	1122	1083	746	1937	229	1564	1084	**1703**	1754	46	1983	346	1643	1519	1707	70	2053
∪\κ	∪	THM	3177	3262	3191	0	3455	89	3142	3191	**3345**	3435	69	3524	225	3338	2892	3419	120	3644
		CSA	0	0	0	**2385**	2385	538	1667	0	1890	1910	63	2448	872	1533	1606	1645	69	2517
		∪	3177	3262	3191	2385	5840	627	4809	3191	**5235**	5345	132	5972	1097	4871	4498	5064	189	6161

better in **S5** (9.3% on average), and even slightly better when averaged over all logics supported by the native modal logic systems (10.1% on average). For disproving conjectures, the embedding approach is significantly stronger overall. This makes the embedding approach the overall stronger approach wrt. the number of problems solved. The evaluation furthermore shows that automating FOML by embedding into THF is superior to using a TFF embedding; only marginally stronger for proving theorems but much stronger for disproving conjectures. This result may be somewhat counter-intuitive as automated reasoning in TFF is much more developed than automated reasoning in THF. The evaluation results further substantiate the effectiveness of the optimizations of **S5** embeddings discussed in Section 4.

In earlier work [45] the embedding approach could not be used for disproving conjectures because it did not ensure that terms are interpreted locally (which is what the QMLTP assumes). This is now properly addressed in the logic embedding tool. This enables model finding using the logic embedding, but also improved reasoning performance for proving theorems.

5.3 Results for Multi-Modal Problems

The numbers of multi-modal problems solved is shown in Table 3. The table is organized as before. The results show that the best performing individual system is Vampire, which solves 19 out of the 20 problems using the TFF embedding. MleanCoP and nanoCoP-M solve 17 out of the 19 non-**K** problems. Due to the inability of the THF-based systems to disprove conjectures, and Nitpick's inability to prove conjectures, the best result of any individual THF-based system is 10 solutions. However, when considering the union problems solved, the THF-based approach is the only one to solve all 20 multi-modal problems. Up to the authors' knowledge the THF embedding approach is the first to ever produce solutions to all the multi-modal problems in the QMLTP.

Table 3: Multi-modal Problems Solved

Language		THF						TFF						T?0		NXF				∪
System		E	Leo	Vam	Nit	∪	\TFF	E	Leo	Vam	∪	\THF	∪	\NXF	MlC	nnC	∪	\T?0	∪
K	THM	1	1	1	0	1	0	1	1	1	1	0	1	1	-	-	-	-	-
K	CSA	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-
K	∪	1	1	1	0	1	0	1	1	1	1	0	1	1	-	-	-	-	-
D	THM	2	2	2	0	2	0	2	2	2	2	0	2	0	2	2	2	0	2
D	CSA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	∪	2	2	2	0	2	0	2	2	2	2	0	2	0	2	2	2	0	2
S4	THM	5	5	5	0	5	0	5	5	5	5	0	5	1	4	4	4	0	5
S4	CSA	0	0	0	7	7	1	5	0	6	6	0	7	0	6	6	6	0	7
S4	∪	5	5	5	7	12	1	10	5	11	11	0	12	1	10	10	10	0	12
S5	THM	2	2	2	0	2	0	2	2	2	2	0	2	0	2	2	2	0	2
S5	CSA	0	0	0	3	3	0	3	0	3	3	0	3	0	3	3	3	0	3
S5	∪	2	2	2	3	5	0	5	2	5	5	0	5	0	5	5	5	0	5
∪	THM	10	10	10	0	10	0	10	10	10	10	0	10	2	8	8	8	0	10
∪	CSA	0	0	0	10	10	1	8	0	9	9	0	10	0	9	9	9	0	10
∪	∪	10	10	10	10	20	1	18	10	**19**	19	0	20	2	17	17	17	0	20
∪**K**	THM	9	9	9	0	9	0	9	9	9	9	0	9	1	8	8	8	0	9
∪**K**	CSA	0	0	0	10	10	1	8	0	9	9	0	10	0	9	9	9	0	10
∪**K**	∪	9	9	9	10	19	1	17	9	**18**	18	0	19	1	17	17	17	0	19

6 Conclusion

The goals of this work were:

1. Test the process of translating first-order modal logic problems into classical first- and higher-order logic using a shallow embedding.
2. Evaluate multiple first- and higher-order ATP systems' abilities to solve the embedded modal logic problems.
3. Compare the performance of native modal logic ATP systems with the embedding approach.

The corresponding findings are:

1. The embedding process is reliable and successful.

2. For some logics, native modal logic systems prove slightly more theorems than using the embedding approach, and for some logics the other way around. Overall the embedding approach proves more theorems than the native modal logic systems

3. The embedding approach outperforms the native modal logic systems for disproving conjectures.

4. The first-order and higher-order systems prove similar numbers of embedded theorems, and the higher-order model finder disproves many more embedded conjectures than the first-order systems.

5. The embedding approach can cope with a wider range of modal logics than the native modal systems considered.

Further work.
As noted in a footnote in Section 3, the development of TPTP World standards for writing ATP solutions beyond common derivations (e.g., CNF refutations) and models (e.g., finite models) is still necessary. In the context of the embedding approach, one particular issue is converting the TFF and THF proofs back to suitable proofs in the original non-classical logic. This remains further work.

Building general support for non-classical logics in the TPTP World is current and ongoing work [43]. The TPTP problem library v9.0.0 released in July 2024 includes modal logic problems. Collecting modal logic problems for the TPTP is ongoing work. In parallel, the tool chains that support use of these problems will be refined and made easily accessible. In particular, the TPTP4X utility [47] will be extended to output formats for existing non-classical ATP systems, to provide those systems with a bridge to the TPTP problems until they adopt the TPTP language natively. Contemporary systems to bridge to include, e.g., $\mathrm{K}_{\mathrm{S}}\mathrm{P}$ [28, 35], nanoCoP-M 2.0 [32], MleanCoP [31], MetTeL2 [54], LoTREC [16], and MSPASS [26].

References

[1] P.B. Andrews. General Models and Extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[2] R. Barcan. A Functional Calculus of First Order Based on Strict Implication. *Journal of Symbolic Logic*, 11:1–16, 1946.

[3] C. Benzmüller. Universal (Meta-)Logical Reasoning: Recent Successes. *Science of Computer Programming*, 172:48–62, 2019.

[4] C. Benzmüller, C.E. Brown, and M. Kohlhase. Higher-order Semantics and Extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004.

[5] C. Benzmüller and D. Miller. Automation of Higher-Order Logic. In D. Gabbay, J. Siekmann, and J. Woods, editors, *Handbook of the History of Logic*, volume 9 - Computational Logic, pages 215–254. North Holland, Elsevier, 2014.

[6] C. Benzmüller, J. Otten, and T. Raths. Implementing and Evaluating Provers for First-order Modal Logics. In L. De Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. Lucas, editors, *Proceedings of the 20th European Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications, pages 163–168. IOS Press, 2012.

[7] C. Benzmüller and L. Paulson. Quantified Multimodal Logics in Simple Type Theory. *Logica Universalis*, 7(1):7–20, 2013.

[8] C. Benzmüller and T. Raths. HOL Based First-order Modal Logic Provers. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 8312 in Lecture Notes in Computer Science, pages 127–136. Springer-Verlag, 2013.

[9] C. Benzmüller and B. Woltzenlogel Paleo. The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics. In S. Kambhampati, editor, *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 936–942. AAAI Press, 2016.

[10] A. Bhayat and G. Reger. A Combinator-based Superposition Calculus for Higher-Order Logic. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12166 in Lecture Notes in Artificial Intelligence, pages 278–296, 2020.

[11] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[12] P. Blackburn, J. van Benthem, and F. Wolther. *Handbook of Modal Logic*. Number 3 in Studies in Logic and Practical Reasoning. Elsevier Science, 2006.

[13] J. Blanchette and T. Nipkow. Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder. In M. Kaufmann and L. Paulson, editors, *Proceedings of the 1st International Conference on Interactive Theorem Proving*, number 6172 in Lecture Notes in Computer Science, pages 131–146. Springer-Verlag, 2010.

[14] T. Braüner and S. Ghilardi. First-order modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 549–620. Elsevier B.V., 2007.

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

[16] L. Fariñas del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci. LoTREC: The Generic Tableau Prover for Modal and Description Logics. In R. Gore, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, number 2083 in Lecture Notes in Artificial Intelligence, pages 453–458. Springer-Verlag, 2001.

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

[18] F. Frege. *Grundgesetze der Arithmetik*. Jena, 1893,1903.

[19] J. Garson. Modal Logic. In E. Zalta, editor, *Stanford Encyclopedia of Philosophy*. Stanford University, 2018.

[20] J. Gibbons and N. Wu. Folding Domain-specific Languages: Deep and Shallow Embeddings (Functional Pearl). In J. Jeuring and M. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, pages 339–347. ACM Press, 2014.

[21] T. Gleißner, A. Steen, and C. Benzmüller. Theorem Provers for Every Normal Modal Logic. In T. Eiter and D. Sands, editors, *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 46 in EPiC Series in Computing, pages 14–30. EasyChair Publications, 2017.

[22] M. Gordon and T. Melham. *Introduction to HOL, a Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[23] J. Harrison. HOL Light: A Tutorial Introduction. In M. Srivas and A. Camilleri, editors, *Proceedings of the 1st International Conference on Formal Methods in Computer-Aided Design*, number 1166 in Lecture Notes in Computer Science, pages 265–269. Springer-Verlag, 1996.

[24] L. Henkin. Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[25] G. Hughes and M. Cresswell. *An Introduction to Modal Logic*. Methuen and Co, 1968.

[26] U. Hustadt and R. Schmidt. MSPASS: Modal Reasoning by Translation and First-Order Resolution. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, number 1847 in Lecture Notes in Artificial Intelligence, pages 67–71. Springer-Verlag, 2000.

[27] S. Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[28] C. Nalon, U. Hustadt, and C. Dixon. KSP: Architecture, Refinements, Strategies and Experiments. *Journal of Automated Reasoning*, 64(3):461–484, 2020.

[29] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[30] H.J. Ohlbach. Semantics Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[31] J. Otten. MleanCoP: A Connection Prover for First-Order Modal Logic. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, number 8562 in Lecture Notes in Artificial Intelligence, pages 269–276, 2014.

[32] J. Otten. The nanoCoP 2.0 Connection Provers for Classical, Intuitionistic and Modal Logics. In A. Das and S. Negri, editors, *Proceedings of the 30th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, number 12842 in Lecture Notes in Artificial Intelligence, pages 236–249. Springer-Verlag, 2021.

[33] J. Otten and G. Sutcliffe. Using the TPTP Language for Representing Derivations in Tableau and Connection Calculi. In B. Konev, R. Schmidt, and S. Schulz, editors, *Proceedings of the Workshop on Practical Aspects of Automated Reasoning, 5th International Joint Conference on Automated Reasoning*, pages 90–100, 2010.

[34] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M. Srivas. PVS: Combining Specification, Proof Checking, and Model Checking. In R. Alur and T.A. Henzinger, editors, *Computer-Aided Verification*, number 1102 in Lecture Notes in Computer Science, pages 411–414. Springer-Verlag, 1996.

[35] F. Papacchini, C. Nalon, U. Hustadt, and C. Dixon. Efficient Local Reductions to Basic Modal Logic. In A. Platzer and G. Sutcliffe, editors, *Proceedings of the 28th International Conference on Automated Deduction*, number 12699 in Lecture Notes in Computer Science, pages 76–92. Springer-Verlag, 2021.

[36] A. N. Prior. Modality and quantification in S5. *J. Symb. Log.*, 21(1):60–62, 1956.

[37] T. Raths and J. Otten. The QMLTP Problem Library for First-Order Modal Logics. In B. Gramlich, D. Miller, and U. Sattler, editors, *Proceedings of the 6th International Joint Conference on Automated Reasoning*, number 7364 in Lecture Notes in Artificial Intelligence, pages 454–461. Springer-Verlag, 2012.

[38] S. Schulz, S. Cruanes, and P. Vukmirović. Faster, Higher, Stronger: E 2.3. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 495–507. Springer-Verlag, 2019.

[39] J. Siekmann, C. Benzmüller, and S. Autexier. Computer Supported Mathematics with OMEGA. *Journal of Applied Logic*, 4(4):533–559, 2006.

[40] A. Steen. An extensible logic embedding tool for lightweight non-classical reasoning (short paper). In B. Konev, C. Schon, and A. Steen, editors, *Proceedings of the 8th Workshop on Practical Aspects of Automated Reasoning*, number 3201 in CEUR Workshop Proceedings, page Online, 2022.

[41] A. Steen. logic-embedding v1.8.4, 2024. DOI: 10.5281/zenodo.10819185.

[42] A. Steen and C. Benzmüller. The Higher-Order Prover Leo-III. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Proceedings of the 8th International Joint Conference on Automated Reasoning*, number 10900 in Lecture Notes in Artificial Intelligence, pages 108–116, 2018.

[43] A. Steen and G. Sutcliffe. TPTP World Infrastructure for Non-classical Logics. In C. Nalon, A. Steen, and M. Suda, editors, *Proceedings of the 9th Workshop on Practical Aspects of Automated Reasoning*, number 3717 in CEUR Workshop Proceedings, pages 74–90, 2024.

[44] A. Steen, G. Sutcliffe, and C. Benzmüller. Supplemental material to "Solving Quantified Modal Logic Problems by Translation to Classical Logics", 2024. DOI: 10.5281/zenodo.10802221.

[45] A. Steen, G. Sutcliffe, T. Scholl, and C. Benzmüller. Solving Modal Logic Problems by Translation to Higher-order Logic. In A. Herzig, J. Luo, and P. Pardo, editors, *Proceedings of the 5th International Conference on Logic and Argumentation*, number 14156 in Lecture Notes in Computer Science, pages 25–43. Springer, 2023. (Best paper award).

[46] A. Stump, G. Sutcliffe, and C. Tinelli. StarExec: a Cross-Community Infrastructure for Logic Solving. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, number 8562 in Lecture Notes in Artificial Intelligence, pages 367–373, 2014.

[47] G. Sutcliffe. TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, *Proceedings of the 2nd International Symposium on Computer Science in Russia*, number 4649 in Lecture Notes in Computer Science, pages 6–22. Springer-Verlag, 2007.

[48] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[49] G. Sutcliffe. The Logic Languages of the TPTP World. *Logic Journal of the IGPL*, 31(6):1153–1169, 2023.

[50] 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.

[51] G. Sutcliffe, L. Kotthoff, C.R. Perrault, and Z. Khalid. An Empirical Assessment of Progress in Automated Theorem Proving. In C. Benzmüller, M. Heule, and R. Schmidt, editors, *Proceedings of the 12th International Joint Conference on Automated Reasoning*, number 14739 in Lecture Notes in Artificial Intelligence, pages 53–74, 2024.

[52] G. Sutcliffe, J. Zimmer, and S. Schulz. TSTP Data-Exchange Formats for Automated Theorem Proving Tools. In W. Zhang and V. Sorge, editors, *Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems*, number 112 in Frontiers in Artificial Intelligence and Applications, pages 201–215. IOS Press, 2004.

[53] V. Thion, S. Cerrito, and M. Cialdea Mayer. A General Theorem Prover for Quantified Modal Logics. In U. Egly and C. Fermüller, editors, *Proceedings of the 11th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, number 2381 in Lecture Notes in Computer Science, pages 266–280. Springer, 2002.

[54] D. Tishkovsky, R. Schmidt, and M. Khodadadi. The Tableau Prover Generator MetTeL2. In L. Fariñas del Cerro, A. Herzig, and J. Mengin, editors, *Proceedings of the 13th European conference on Logics in Artificial Intelligence*, number 7519 in Lecture Notes in Computer Science, pages 492–495. Springer, 2012.

[55] H. van Ditmarsch, J. Halpern, W. van der Hoek, and B. Kooi. *Handbook of Epistemic Logic*. College Publications, 2015.

[56] P. Vukmirović, A. Bentkamp, J. Blanchette, S. Cruanes, V. Nummelin, and S. Tourret. Making Higher-order Superposition Work. In A. Platzer and G. Sutcliffe, editors, *Proceedings of the 28th International Conference on Automated Deduction*, number 12699 in Lecture Notes in Computer Science, pages 415–432. Springer-Verlag, 2021.

Author information

1. Alexander Steen
   University of Greifswald, Germany
   `alexander.steen@uni-greifswald.de`
   ORCID: 0000-0001-8781-9462

2. Geoff Sutcliffe
   University of Miami, USA

geoff@cs.miami.edu
ORCID: 0000-0001-9120-3927

3. Christoph Benzmüller
University of Bamberg and FU Berlin, Germany
christoph.benzmueller@uni-bamberg.de
ORCID: 0000-0002-3392-30935

A Detailed Example of a FOML Embedding

Assuming a constant domain, **S5** modal logic with rigid symbols and local terms, the FOML example from Section 4 is represented in TPTP format as follows:

```
tff(simple_spec,logic,
    $modal == [
      $domains == $constant,
      $designation == $rigid,
      $terms == $local,
      $modalities == $modal_system_S5 ] ).

tff(barcan, conjecture,
  ( ![X:$i]: ( {$box} @ (p(X)) ) )
    => ( {$box} @ (![X:$i]: (p(X))) ) ).
```

The resulting first-order embedding in TFF is then given by:

```
tff('$world_type',type,
    '$world': $tType ).

tff('$local_world_decl',type,
    '$local_world': '$world' ).

tff('$accessible_world_decl',type,
    '$accessible_world': ( '$world' * '$world' ) > $o ).

tff(mrel_universal,axiom,
    ! [W: '$world',V: '$world'] : '$accessible_world'(W,V) ).

tff(barcan,conjecture,
    ( ! [X: $i,W: '$world'] :
        ( '$accessible_world'('$local_world',W)
       => p(W,X) )
   => ! [W: '$world'] :
        ( '$accessible_world'('$local_world',W)
       => ! [X: $i] : p(W,X) ) ) ).
```

The analogous higher-order embedding in THF is given by (note that β-normalization is not done by the embedding tool):

```
thf(mworld_type,type, mworld: $tType ).
thf(mrel_decl,type, mrel: mworld > mworld > $o ).
thf(mbox_decl,type, mbox: ( mworld > $o ) > mworld > $o ).
thf(mdia_decl,type, mdia: ( mworld > $o ) > mworld > $o ).
thf(mactual_decl,type, mactual: mworld ).
thf(mlocal_decl,type, mlocal: ( mworld > $o ) > $o ).
thf(p_type,type, p: mworld > $i > $o ).

thf(mbox_def,definition,
    ( mbox
    = ( ^ [Phi: mworld > $o,W: mworld] :
        ! [V: mworld] :
          ( ( mrel @ W @ V )
         => ( Phi @ V ) ) ) ) ).

thf(mdia_def,definition,
    ( mdia
    = ( ^ [Phi: mworld > $o,W: mworld] :
        ? [V: mworld] :
          ( ( mrel @ W @ V )
          & ( Phi @ V ) ) ) ) ).

thf(mrel_universal,axiom, ! [W: mworld,V: mworld] : ( mrel @ W @ V ) ).

thf(mlocal_def,definition,
    ( mlocal = ( ^ [Phi: mworld > $o] : ( Phi @ mactual ) ) ) ).

thf(barcan,conjecture,
    ( mlocal
    @ ^ [W: mworld] :
        ( ( ^ [W: mworld] :
            ! [X: $i] :
              ( mbox
              @ ^ [W: mworld] :
                  ( p @ W
                  @ ( ^ [W: mworld] : X
                    @ W ) )
              @ W )
          @ W )
       => ( mbox
          @ ^ [W: mworld] :
            ! [X: $i] :
              ( ^ [W: mworld] :
                  ( p @ W
                  @ ( ^ [W: mworld] : X
                    @ W ) )
              @ W )
          @ W ) ) ) ).
```