The MET: The Art of Flexible Reasoning with Modalities

Tobias Gleißner(✉) and Alexander Steen

Institute of Computer Science, Freie Universität Berlin, Berlin, Germany
{tobias.gleissner,a.steen}@fu-berlin.de

Abstract. Modal logics have numerous applications in computational linguistics, artificial intelligence, rule-based reasoning, and, in general, alethic, deontic and epistemic contexts. Higher-order quantified modal logics additionally incorporate the expressiveness of higher-order formalisms and thereby provide a quite general reasoning framework. By exploiting this expressiveness, the Modal Embedding Tool (MET) allows to automatically encode higher-order modal logic problems into equivalent problems of classical logic, enabling the use of a broad variety of established reasoning tools. In this system description, the functionality and usage of MET as well as a suitable input syntax for flexible reasoning with modalities are presented.

1 Introduction

Various powerful automated and interactive theorem proving systems (ATP and ITP, respectively) for first-order (FO) and higher-order (HO) logics have been developed over the past decades, including the first-order ATP E [1], the higher-order ATPs Satallax [2], LEO-II [3] and Leo-III [4], and the higher-order ITP Isabelle/HOL [5]. While many of these systems are meanwhile quite robust and mature, they often support reasoning in classical logics only. This is in contrast to the fact that non-classical logics have many topical applications in mathematics, computer science and beyond. In this work, we focus on the automation of quantified (multi-)modal logics [6] which can be fruitfully applied in the context of artificial intelligence, computational linguistics and rule-based reasoning. They also play an important role in various areas of philosophy, including ontology, (computer-)ethics, philosophy of mind and philosophy of science. Many challenging applications, however, as recently explored in metaphysics [7–9], require quantified and in particular higher-order quantified modal logics (HOMLs). But even for first-order non-classical logics only a small number of implemented systems is available to date, and the situation is even worse for higher-order quantified logics. In particular, the development of ATPs for HOMLs is still in its infancy, hence impeding more complex computer-assisted studies of relevant topics.

To overcome this situation, in this work we present the **M**odal **E**mbedding **T**ool (MET for short) that bridges the above gap by enabling the employment

Alexander Steen is partially supported by the Volkswagenstiftung (project CRAP).

of classical higher-order reasoning systems, including powerful HO ATPs, for reasoning in a broad variety of quantified modal logics. Recall that for quantified modal logics there exist multiple different notions of semantics, most of which usually used in different application domains. The exact semantics of a given quantified modal logic can be regarded a product of multiple individual semantical parameters, including:

(i) Modal axiomatization: *What properties hold for each modality?*
The properties range from axiom scheme K alone to the strong assumptions of logic S5, and any intermediate system (cf. the modal logic cube [6]).
(ii) Quantification semantics: *What are the domains of quantified variables?*
Usual choices include so-called cumulative, decreasing, constant and varying domain semantics.
(iii) Rigidity: *Is the meaning of a symbol the same in every possible world?*
Possible choices include rigid and world-dependent constant symbols.

Also, there exist different choices for logical consequence relations, including at least so-called local and global consequence [6]. When taking all possible parameter combinations into account this amounts to more than 120 different HOMLs.

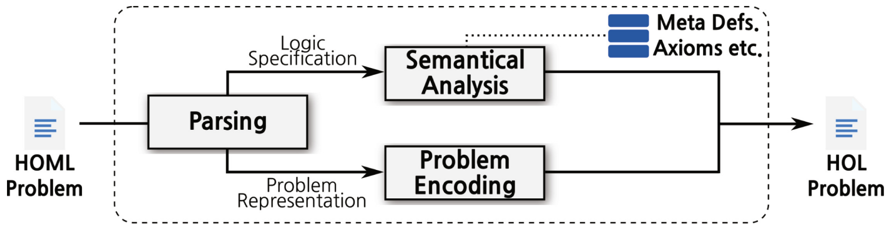

Fig. 1. Working principle of the MET: A modal logic problem statement is given to the system which it then transforms and augments with suitable technical definitions. The resulting problem is formulated in classical logic and only valid if the original problem was valid.

MET implements a shallow semantical embedding approach [10] in which formulas of modal logic are identified with specific terms of classical higher-order logic such that a notion of modal validity can be defined within HOL that coincides with the desired modal logic semantics. More concretely, a modal logic formulated in a suitable machine-readable syntax (cf. Figs. 2 and 3) is translated by MET to a problem statement in the de-facto standard TPTP THF syntax [11,12] for HO ATP systems. The transformation process is thereby validity preserving and thus allows the use of common reasoning systems that do not support modal logic reasoning on their own. The process is visualized in Fig. 1. Although the here discussed approach involves an indirection over classical HOL, recent evaluations confirm that the reasoning effectivity of HO ATPs used in conjunction with MET are on a par with established modal logic reasoners [4].

Additionally, reasoning using MET is much more flexible than the employment of these special purpose systems as it allows a completely free choice of all relevant semantical parameters (supporting every existent normal modal logic) whereas native modal logic reasoners are limited to a small subset of modal logic systems and usually have fixed choices for quantification semantics, symbol rigidity and consequence. In fact, even more general modal logics are supported via a fine-grained control over the properties of individual quantification domains, modal operators, etc [13].

Whereas earlier work focused on the theoretical foundations [10], and the development of the automatic embedding procedure itself [13], in this paper, we present the prover-independent tool MET and its practical employment in relevant application scenarios.

Higher-Order Modal Logics. HOMLs as addressed here are extensions of HOL [14]. HOL provides λ-abstraction as an elegant means to denote unnamed functions, predicates and sets (by their characteristic functions). HOML, in turn, augments HOL with a set of modal operators $\Box^i$, $i \in I$, for some index set I, and is equipped with a suitable combination of HOL semantics and a Kripke-style modal semantics [13]. In our approach an adequate notion of Henkin semantics for both HOML and HOL is assumed [10,13].

2 A Syntax for HOML and Its Semantics

A standard ASCII-based machine-readable representation of HOL problems for ATP systems is given by the TPTP THF dialect [12]. This syntax is supported by most of the current HOL reasoners, including all HO systems mentioned in the beginning of Sect. 1. Since the syntax of HOML is a conservative extension of that of classical HOL, the THF representation language can as well easily be augmented by introducing the modal operators `$box` and `$dia` as new primitive connectives, representing the modal connectives $\Box$ and $\Diamond$, respectively (in a mono-modal settings). For multi-modal logics, there exist analogous operators that are additionally given an index as first argument and the formula as second argument. The remaining syntax coincides with standard THF and is described in the literature [12].

As sketched in Sect. 1, there is no single semantics of quantified modal logics. As a consequence, there is additional need for explicitly stating the semantical setting in which a problem is to be assessed by a reasoning system. This is realized in the here proposed syntax by including a meta-logical specification into the problem header. Such a specification statement is displayed in Fig. 2: In this logic specification, the identifiers `$constants`, `$quantification` and `$consequence` specify the exact semantical settings for the rigidity of constant symbols, the quantification semantics and the consequence relation, respectively. Finally, `$modalities` specifies the properties of the modal connectives by means of fixed modal logic system names or, alternatively, a list of individual modal axiom schemes. The valid parameter values are given in Table 1.

```
% Begin of logic specification
thf(⟨name₀⟩, logic, ($modal := [
      $constants := ⟨const_spec⟩, $quantification := ⟨domain_spec⟩,
      $consequence := ⟨conseq_spec⟩, $modalities := ⟨modal_spec⟩ ] )).
% End of logic specification, begin of problem statement
thf(⟨name₁⟩, ⟨role₁⟩, ⟨formula₁⟩).
...
thf(⟨nameₙ⟩, ⟨roleₙ⟩, ⟨formulaₙ⟩).
```

Fig. 2. Layout of a general modal logic problem. The first statement (ll. 2–4) specifies a concrete modal logic, the remaining statements (ll. 6–8) formulate the problem itself. The $\langle \mathsf{name}_i \rangle$ serve as syntactic identifier for that statement, a $\langle \mathsf{role}_i \rangle$ (usually set to `axiom` or `conjecture`) tells the reasoning system how to interpret the $\langle \mathsf{formula}_i \rangle$ formulated in the presented augmented THF syntax. Lines starting with `%` are comments.

Table 1. Semantic specification parameters. The parameter placeholders, written in angles $\langle \cdot \rangle$, refer to the values for the logic specification of Fig. 2. The names of the modal logic system parameters (such as `$modal_system_K` or `$modal_system_S5`) refer to the respective systems from the modal logic cube [6]. The individual modal axiom schemes names (such as `$modal_axiom_T` or `$modal_axiom_5`) are named similarly.

Parameter	**Valid values**
⟨const_spec⟩	`$rigid`, `$flexible`
⟨domain_spec⟩	`$constant`, `$varying`, `$cumulative`, `$decreasing`
⟨conseq_spec⟩	`$local`, `$global`

⟨modal_spec⟩	`$modal_system_X`
	for `X` in {`K`, `KB`, `K4`, `K5`, `K45`, `KB5`, `D`, `DB`, `D4`, `D5`, `D45`, `T`, `B`, `S4`, `S5`, `S5U`}
	or
	[`$modal_axiom_X`$_1$, `$modal_axiom_X`$_2$, ...]
	for `X`$_i$ in {`T`, `B`, `D`, `4`, `5`, `CD`, `C4`, `C`}

The remaining placeholders of Fig. 2, ⟨name⟩, ⟨role⟩ and ⟨formula⟩, are standard and given by the TPTP language definition [11] to which we refer to for brevity. The semantics specification format presented in this paper is work-in-progress and stems from an ongoing TPTP language extension proposal.[1]

3 Application Examples

In this section, the practical employment of MET for reasoning with relevant non-trivial problem statements is discussed. The first application example, a formulation of the wise men puzzle, incorporates the use of multiple inter-related modality operators and quantification beyond first-order. The second example

[1] See proposal "Logic Specification Format" of the TPTP platform for more details.

focuses on the use of logic specification statements within the problem and illustrates the flexibility of the here presented reasoning approach.

3.1 Case Study: The Wise Men Puzzle

A classical example dealing with knowledge between agents and implicit knowledge transfer is the wise men puzzle (also known in a variation as muddy forehead puzzle). Epistemic logic, the logic about knowledge, can be interpreted as a form of multi-modal logic, were the modality operators represent *knowing* and are indexed with an agent's identifier from an index set I (referring to the particular agent whose knowledge it addresses). As an example, the sentence "agent a knows ϕ", for an agent $a \in I$, can be stated as $\Box^a\phi$. While dealing with common knowledge scenarios, often an additional artificial agent (sometimes referred to as *fool*) is defined for allowing statements such as "everybody knows ϕ", represented by $\Box^{\text{fool}}\phi$.

```
thf(wise_men_puzzle_semantics, logic , ( $modal := [
    $constants := $rigid,  $quantification := $varying,
    $consequence := $global, $modalities := $modal_system_S5] )).

% $i type models the agents's hats
thf(agent_a, type, (a: $i)).
thf(agent_b, type, (b: $i)).
thf(agent_c, type, (c: $i)).

% Property of an agent's hat: ws represents "having a white spot"
thf(white_spot, type, (ws: ($i>$o))).

% Common knowledge: At least one agent has a white spot
thf(axiom_1, axiom, ($box_int @ 0 @ ((ws @ a) | (ws @ b) | (ws @ c)))).

% If one agent has a white spot all other agents can see this
thf(axiom_2ab, axiom, ($box_int @ 0 @ ((ws @ a) => ($box_int @ 2 @ (ws @ a))))).
thf(axiom_2ac, axiom, ($box_int @ 0 @ ((ws @ a) => ($box_int @ 3 @ (ws @ a))))).
thf(axiom_2ba, axiom, ($box_int @ 0 @ ((ws @ b) => ($box_int @ 1 @ (ws @ b))))).
thf(axiom_2bc, axiom, ($box_int @ 0 @ ((ws @ b) => ($box_int @ 3 @ (ws @ b))))).
thf(axiom_2ca, axiom, ($box_int @ 0 @ ((ws @ c) => ($box_int @ 1 @ (ws @ c))))).
thf(axiom_2cb, axiom, ($box_int @ 0 @ ((ws @ c) => ($box_int @ 2 @ (ws @ c))))).

% If one agent has a black spot all other agents can see this
thf(axiom_3ab, axiom, ($box_int @ 0 @ ((~(ws @ a)) => ($box_int @ 2 @ (~(ws @ a)))))).
thf(axiom_3ac, axiom, ($box_int @ 0 @ ((~(ws @ a)) => ($box_int @ 3 @ (~(ws @ a)))))).
thf(axiom_3ba, axiom, ($box_int @ 0 @ ((~(ws @ b)) => ($box_int @ 1 @ (~(ws @ b)))))).
thf(axiom_3bc, axiom, ($box_int @ 0 @ ((~(ws @ b)) => ($box_int @ 3 @ (~(ws @ b)))))).
thf(axiom_3ca, axiom, ($box_int @ 0 @ ((~(ws @ c)) => ($box_int @ 1 @ (~(ws @ c)))))).
thf(axiom_3cb, axiom, ($box_int @ 0 @ ((~(ws @ c)) => ($box_int @ 2 @ (~(ws @ c)))))).

% Agents 1 and 2 do not know their hat color
thf(axiom_9, axiom, ($box_int @ 0 @ (~($box_int @ 1 @ (ws @ a))))).
thf(axiom_10, axiom, ($box_int @ 0 @ (~($box_int @ 2 @ (ws @ b))))).

% Agent 3 can deduce the color of his hat (white spot)
thf(con, conjecture, ($box_int @ 3 @ (ws @ c))).
```

Fig. 3. The wise men puzzle formulated in modal THF syntax. The term `$box_int @ i` represents a box operator $\Box^i$ for which the set of integers serves as index set I. In this example, the common knowledge agent (the *fool*) is given by index 0, the remaining three agents by indexes $1, 2$ and 3.

A formulation of the wise men puzzle is given in Fig. 3. In the logic specification, the modalities (including the common knowledge modality) are given an S5 axiomatization to capture the usual assumptions about knowledge. Additionally, a varying domain semantics is used for this experiment. The modal operators $\Box^a$ for some agent $a \neq \text{fool} \in I$ are related to common knowledge $\Box^{\text{fool}}$ using so-called bridge-rules stating that everything that is common knowledge is also known by the individual agents (cf. ll. 16–30). The common knowledge fact that the first two agents do not know their hat color is given by two axioms (ll. 33–34) and finally the conjecture that the third agent now knows its hat color is given by the conjecture (l. 37). The wise men problem in the presented formulation can be solved using MET in conjunction with Leo-III as reasoner back end in under 5s, cf. Appendix A for a detailed display of the tools usage.

3.2 Case Study: Experiments with Semantical Variations

In this case study, we focus on the flexibility the logic specification within a problem provides for experiments in different semantical settings. Figure 4 displays an example modal logic formula that is an instance of a corollary of Becker's postulate [15]. It essentially expresses that everything that is possibly necessary it, in fact, necessary. Since this formula is obviously debatable, one might want to explicitly include or exclude this fact from a logical system. It is known from the literature, that Becker's postulate is indeed valid in S5 modal logics but not in any weaker logic systems. Even without this knowledge, the MET allows to experimentally reproduce these results with only simple modification of the logic specification statements. To that end, each semantical setting can be formulated as logic specification and then transformed by MET to HOL problems. These HOL problems are then in turn given to HO reasoning systems for verifying or refuting the conjecture.

```
thf(s5_spec, logic, ($modal := [
      $constants := $rigid, $quantification := $constant,
      $consequence := $global, $modalities := $modal_system_S5 ])).
thf(becker,conjecture,( ! [P:$i>$o,F:$i>$i, X:$i]: (? [G:$i>$i]:
      (($dia @ ($box @ (P @ (F @ X))) => ($box @ (P @ (G @ X))))))).
```

Fig. 4. A corollary of Becker's postulate formulated in modal THF, representing the formula $\forall P_{\iota\to o}\forall F_{\iota\to\iota}\forall X_\iota\exists G_{\iota\to\iota}(\Diamond\Box P(F(X)) \Rightarrow \Box P(G(X)))$.

In the example of Becker's postulate, the higher-order ATP system Leo-III and the counter-model finder Nitpick [16] verify the above claim. The systems produce proofs resp. explicit, finite, counter-models of the validity the conjecture in each modal logic system. The results of these experiments are summarized in Table 2. It can be seen that, for every modal logic system, the combination of both reasoners successfully assess the conjecture and yield the expected results.

Each invocation of the reasoning systems (including the pre-processing by MET) takes less than 1 s. Note that both systems are in a sense complementary, i.e. theorem proving systems are usually stronger for proving the validity of a conjecture while counter-model finders focus on counter-satisfiability. Using both systems, positive and negative results can be established as desired.

The example of Becker's postulate is chosen for demonstrative purposes. Similarly interesting formulas for certain modal logics such as Barcan's formula (or its converse) can be analyzed analogously using MET [13]. In a more general setting, the semantical flexibility of the here presented approach allows for an empirical assessment of a formal system's adequateness for a specific application; and to explore further, possibly unintended, consequences of a given formulation.

Table 2. Evaluation results of the validity of Becker's postulate from Fig. 4. For each semantical setting, the factual validity of the postulate (Expected) and the actual results of Leo-III and Nitpick (Result) are presented. ✓ and × denote validity resp. invalidity of the postulate under the respective semantics as well as a system's according result. A timeout of a system (i.e. no feasible result) is denoted †. Quantification semantics are abbreviated co and va for constant and varying domains, respectively.

(a) Leo-III

Modal System	K		B		T		S4		S5	
Domains	co	va	co	va	co	va	co	va	co	va
Expected	×	×	×	×	×	×	×	×	✓	✓
Result	†	†	†	†	†	†	†	†	✓	✓

(b) Nitpick

Modal System	K		B		T		S4		S5	
Domains	co	va	co	va	co	va	co	va	co	va
Expected	×	×	×	×	×	×	×	×	✓	✓
Result	×	×	×	×	×	×	×	×	†	†

4 Summary and Further Work

In this work, a self-contained syntax for formulating higher-order modal logic problems was sketched that is used as input format to the Modal Embedding Tool. This stand-alone tool acts as external pre-processor for HO reasoning systems and emits for a given input problem statement an equivalent (wrt. validity) HOL problem formulated in standard THF syntax. MET is implemented in Java and freely available at GitHub under BSD-3 license.[2] The higher-order ATP system Leo-III additionally incorporates a version of MET for automatically embedding modal logic problems without any need for external pre-processing.

When used in conjunction with further powerful HO ATP systems, MET has many topical applications for reasoning in knowledge bases, legal reasoning, smart contracts and, more generally, in alethic, epistemic and deontic contexts. An adaption of MET for accepting RuleML input syntax [17], OWL [18] or further languages for rule-based reasoning is, thanks to the flexible underlying embedding approach, straight-forward and current work-in-progress [19]. The MET can also be extended to serve as a translation tool between these different representation formats.

[2] See github.com/leoprover/embed_modal for details and further instructions.

A Installation and Usage of MET

Acquisition and Installation

MET is freely available on GitHub (https://github.com/leoprover/embed_modal) under BSD-3 license. The most current release is always accessible under https://github.com/leoprover/embed_modal/releases/latest. To get it, simply download the source archive and extract it so some location.

```
> wget https://github.com/leoprover/embed_modal/archive/1.0.tar.gz
> tar -xvzf 1.0.tar.gz
```

After extraction, MET can be built using Make. Simply `cd` to the extracted directory s and run Make:

```
> cd embed_modal-1.0
> make
```

After building, there should be a directory `bin/`, relative from the current directory. This directory contains the binary `embedlogic` of MET. You will also find a JAR in the directory `embed/target/` which you can use as a library for your own projects.

MET can optionally be installed by invoking

```
> make install
```

which copies the binary to the directory `$HOME/.local/bin` and adds it to your `$PATH`.

Usage

To execute MET, simply run the `embedlogic` command (assuming you have installed MET) or run `bin/embedlogic`. For brevity, we assume that `embedlogic` is available.

For the example of Becker's postulate, running

```
> embedlogic -i becker.p -o becker_embedded.p
```

will generate a new file `becker_embedded.p` that contains the embedded THF problem that is semantically equivalent to the modal problem of `becker.p` as given in Fig. 4 (the file is also contained in the distribution of MET in `examples/`). Now, any TPTP THF-compliant ATP system can be used, e.g. Leo-III can be invoked on the result:

```
> leo3 becker_embedded.p
% Axioms used in derivation (1): mrel_meuclidean
[...]
% SZS status Theorem for becker.p : 3443 ms resp. 1260 ms w/o parsing
```

Becker's Postulate Embedded

The embedded file `becker_embedded.p` contains the following:

```
% declare type for possible worlds
thf(mworld_type,type,(
    mworld: $tType )).

% declare accessibility relations
thf(mrel_type,type,(
    mrel: mworld > mworld > $o )).

% define accessibility relation properties
thf(mreflexive_type,type,(
    mreflexive: ( mworld > mworld > $o ) > $o )).

thf(mreflexive_def,definition,
    ( mreflexive
    = ( ^ [R: mworld > mworld > $o] :
        ! [A: mworld] :
          ( R @ A @ A ) ) )).

thf(meuclidean_type,type,(
    meuclidean: ( mworld > mworld > $o ) > $o )).

thf(meuclidean_def,definition,
    ( meuclidean
    = ( ^ [R: mworld > mworld > $o] :
        ! [A: mworld,B: mworld,C: mworld] :
          ( ( ( R @ A @ B )
            & ( R @ A @ C ) )
         => ( R @ B @ C ) ) ) )).

% assign properties to accessibility relations
thf(mrel_mreflexive,axiom,(
    mreflexive @ mrel )).

thf(mrel_meuclidean,axiom,(
    meuclidean @ mrel )).

% define valid operator
thf(mvalid_type,type,(
    mvalid: ( mworld > $o ) > $o )).

thf(mvalid_def,definition,
    ( mvalid
    = ( ^ [S: mworld > $o] :
        ! [W: mworld] :
          ( S @ W ) ) )).

% define nullary, unary and binary connectives which are no quantifiers
thf(mimplies_type,type,(
    mimplies: ( mworld > $o ) > ( mworld > $o ) > mworld > $o )).

thf(mimplies,definition,
    ( mimplies
    = ( ^ [A: mworld > $o,B: mworld > $o,W: mworld] :
          ( ( A @ W )
         => ( B @ W ) ) ) )).

thf(mdia_type,type,(
    mdia: ( mworld > $o ) > mworld > $o )).

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

thf(mbox_type,type,(
    mbox: ( mworld > $o ) > mworld > $o )).

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

% define exists quantifiers
thf(mexists_const_type__o__d_i_t__d_i_c_,type,(
    mexists_const__o__d_i_t__d_i_c_: ( ( $i > $i ) > mworld > $o ) > mworld > $o )).

thf(mexists_const__o__d_i_t__d_i_c_,definition,
    ( mexists_const__o__d_i_t__d_i_c_
    = ( ^ [A: ( $i > $i ) > mworld > $o,W: mworld] :
        ? [X: $i > $i] :
          ( A @ X @ W ) ) )).
```

```
% define for all quantifiers
thf(mforall_const_type__o__d_i_t__o_mworld_t__d_o_c__c_,type,(
    mforall_const__o__d_i_t__o_mworld_t__d_o_c__c_: ( ( $i > mworld > $o ) > mworld > $o ) > mworld > $o )).

thf(mforall_const__o__d_i_t__o_mworld_t__d_o_c__c_,definition,
    ( mforall_const__o__d_i_t__o_mworld_t__d_o_c__c_
    = ( ^ [A: ( $i > mworld > $o ) > mworld > $o,W: mworld] :
        ! [X: $i > mworld > $o] :
          ( A @ X @ W ) ) )).

thf(mforall_const_type__o__d_i_c_,type,(
    mforall_const__o__d_i_c_: ( $i > mworld > $o ) > mworld > $o )).

thf(mforall_const__o__d_i_c_,definition,
    ( mforall_const__o__d_i_c_
    = ( ^ [A: $i > mworld > $o,W: mworld] :
        ! [X: $i] :
          ( A @ X @ W ) ) )).

thf(mforall_const_type__o__d_i_t__d_i_c_,type,(
    mforall_const__o__d_i_t__d_i_c_: ( ( $i > $i ) > mworld > $o ) > mworld > $o )).

thf(mforall_const__o__d_i_t__d_i_c_,definition,
    ( mforall_const__o__d_i_t__d_i_c_
    = ( ^ [A: ( $i > $i ) > mworld > $o,W: mworld] :
        ! [X: $i > $i] :
          ( A @ X @ W ) ) )).

% ------------------------------------------------------------------------
% transformed problem
% ------------------------------------------------------------------------

thf(1,conjecture,
    ( mvalid
    @ ( mforall_const__o__d_i_t__o_mworld_t__d_o_c__c_
      @ ^ [P: $i > mworld > $o] :
          ( mforall_const__o__d_i_t__d_i_c_
          @ ^ [F: $i > $i] :
              ( mforall_const__o__d_i_c_
              @ ^ [X: $i] :
                  ( mexists_const__o__d_i_t__d_i_c_
                  @ ^ [Q: $i > $i] :
                      ( mimplies @ ( mdia @ ( mbox @ ( P @ ( F @ X ) ) ) ) @ ( mbox @ ( P @ ( Q @ X ) ) ) ) ) ) ) ) ) )).
```

References

1. Schulz, S.: E – a brainiac theorem prover. AI Commun. **15**(2,3), 111–126 (2002)
2. Brown, C.E.: Satallax: An automatic higher-order prover. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 111–117. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_11
3. Benzmüller, C., Sultana, N., Paulson, L.C., Theiß, F.: The higher-order prover LEO-II. J. Autom. Reason. **55**(4), 389–404 (2015)
4. Steen, A., Benzmüller, C.: The higher-order prover Leo-III. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNAI, vol. 10900, pp. 108–116. Springer, Heidelberg (2018)
5. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science, vol. 2283. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45949-9
6. Blackburn, P., van Benthem, J.F., Wolter, F.: Handbook of Modal Logic, vol. 3. Elsevier, Amsterdam (2006)
7. Benzmüller, C., Woltzenlogel Paleo, B.: The inconsistency in Gödel's ontological argument: a success story for AI in metaphysics. In: Kambhampati, S. (ed.) IJCAI 2016, vol. 1–3, pp. 936–942. AAAI Press (2016). (Acceptance rate $\leq$ 25%)
8. Benzmüller, C., Weber, L., Woltzenlogel Paleo, B.: Computer-assisted analysis of the Anderson-Hájek controversy. Logica Universalis **11**(1), 139–151 (2017)
9. Fuenmayor, D., Benzmüller, C.: Types, Tableaus and Gödel's God in Isabelle/HOL. Archive of Formal Proofs (2017). This publication is machine verified with Isabelle/HOL, but only mildly human reviewed

10. Benzmüller, C., Paulson, L.: Quantified multimodal logics in simple type theory. Logica Universalis **7**(1), 7–20 (2013). (Special Issue on Multimodal Logics)
11. Sutcliffe, G.: The TPTP problem library and associated infrastructure. From CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017)
12. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. J. Formaliz. Reason. **3**(1), 1–27 (2010)
13. Gleißner, T., Steen, A., Benzmüller, C.: Theorem provers for every normal modal logic. In: Eiter, T., Sands, D. (eds.) LPAR-21. EPiC Series in Computing, Maun, Botswana, vol. 46, pp. 14–30. EasyChair (2017)
14. Andrews, P.: Church's type theory. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Stanford University, Metaphysics Research Lab (2014)
15. Becker, O.: Zur Logik der Modalitäten. Max Niemeyer Verlag (1930)
16. Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 131–146. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14052-5_11
17. Athan, T., Boley, H., Paschke, A.: Ruleml 1.02: Deliberation, reaction and consumer families. In: Bassiliades, N., et al. (eds.) Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015). CEUR Workshop Proceedings, vol. 1417. CEUR-WS.org (2015)
18. Cao, S.T., Nguyen, L.A., Szalas, A.: The web ontology rule language OWL 2 RL^{+} and its extensions. Trans. Comput. Collect. Intell. **13**, 152–175 (2014)
19. Boley, H., Benzmüller, C., Luan, M., Sha, Z.: Translating higher-order modal logic from RuleML to TPTP. In Giurca, A., et al. (eds.) Proceedings of the RuleML 2016 Challenge, the Special Industry Track and the RuleML 2016 Doctoral Consortium hosted by the 10th International Web Rule Symposium (RuleML 2016). CEUR Workshop Proceedings, vol. 1620. CEUR-WS.org (2016)