TPTP And Beyond: Representation of Quantified Non-Classical Logics*

Max Wisniewski[1], Alexander Steen[1], and Christoph Benzmüller[21]

[1] Freie Universität Berlin, Institute of Computer Science
{m.wisniewski,a.steen}@fu-berlin.de
[2] Stanford University, CSLI
c.benzmueller@gmail.com

Abstract

The practical employment of automated deduction systems requires the user to input problem statements in a well-formed string representation. While this presentation is usually fixed by the respective system, the various language dialects of the TPTP library are meanwhile accepted as a de-facto standard for all current automated theorem provers based on classical logics. In the context of reasoning in non-classical logics, however, only a few limited standardization approaches exist, with QMLTP being the most notable exception. To move standardization forward, we outline conservative extensions to the TPTP language that allow systematic syntax definitions for various expressive, non-classical logics. These logics include higher-order versions of modal logics, conditional logics, hybrid logics, free logics, and many-valued logics. We are convinced that a standard syntax for prominent non-classical logics will not only facilitate their deployment but also support the development and comparability of corresponding theorem proving systems.

1 Introduction

Computer-assisted reasoning in non-classical logics is of increasing interest to enable and support applications in e.g. computer science, mathematics and philosophy. Several powerful automated and interactive theorem proving systems have been developed over the past decades. However, when it comes to quantified logics, most of the available systems focus on classical logic only. Amongst the notable exceptions is MleanCoP [26] which automates first-order modal logic.

Orthogonal to the development of specialized provers, the semantical embedding approach [7] allows for a quick adaptation of existing higher-order reasoning systems to a broad variety of expressive, non-classical logics. In fact, for each logic discussed in this paper, we already have a new theorem provers in place [7, 5, 9, 8, 32]. These reasoners have been implemented by utilizing the embeddings approach on top of systems such as Leo-II [4] or Isabelle/HOL [22]. Recent experiments show that this approach indeed offers a surprisingly effective automation of the embedded non-classical logics. However, from the users perspective the utilization of the embeddings approach can become rather involved and distracting. Hence, system users may eventually not want to be exposed to the embeddings at all. Moreover, a comprehensive evaluation of systems based on the embeddings approach against systems based on the direct approach is currently hardly feasible. One reason, in addition to the fact the very few systems in the direct approach are available to date, is the lack of commonly agreed input formats.

In order to amplify the practical development, deployment and comparison of automated reasoning in quantified non-classical logics, we therefore outline problem representation formats for various (quantified) non-classical logics, primarily for use in automated

*This work has been supported by the DFG under grant BE 2501/11-1 (Leo-III) and grant BE 2501/9-2 (Computational Metaphysics).

theorem proving (ATP) systems. More specifically, we present conservative extensions to the well-known TPTP [34] syntax representations. We display proposals on how to represent logical problems in quantified versions of multi-modal logics, hybrid logics, conditional logics, free logics and some propositional many-valued logics. Additionally, we include means of adding meta-logical information to the problem statement that specifies details regarding the assumed semantics of the respective problem and logic. We briefly introduce each of the mentioned logics and describe the necessary modifications of the already existing languages.

TPTP and QMLTP. The *Thousands of Problems for Theorem Provers* problem library (TPTP) [34] provides a coherent environment for testing automated theorem provers for their correctness and performance. To that end, it postulates a standardized and stable formula representation syntax for most classical logic languages (e.g. FOF for first-order formulas or THF [35] for typed higher-order formulas). We will base our problem representation format on the THF dialect and moderately extend the existing syntax definitions to match the requirements of the particular non-classical logic in question.

A closely related project is QMLTP [27] which provides a syntax and a collection of problems for first-order modal logic. The QMLTP syntax is designed as an extension of the TPTP FOF language, introducing special symbols for the box and the diamond operators of modal logic (cf. §3.1). Regarding modal logics (or logics based on modal logic), we will also re-use existing syntax representations of the QMLTP project. Moreover, we adopt and extend the QMLTP approach for the specification of meta-logical information.

2 Classical Higher-Order Logic

We primarily address quantified versions of non-classical logics in this paper. Since we do not intend to (artificially) restrict these logics to be first-order only, their representation formats will quite naturally be given as extensions of (classical) higher-order logic (HOL) [13]. The syntax and semantics of HOL is now briefly introduced as it serves as a basis for later, when logic-specific definitions of the syntax and semantics of non-classical higher-order logics are depicted. The brief introduction to HOL is mainly borrowed from [33] which, in turn, adapts the simplified notation of [21] for HOL.

HOL is a typed logic. The set of *simple types* $\mathcal{T}$ contains all types that are freely generated using the binary function type constructor $\to$ and a set of base types, usually chosen to be $\{o, \iota\}$ for Booleans and individuals, respectively. Terms of HOL are given by the following grammar:

$$s, t ::= c_\tau \mid X_\tau \mid (\lambda X_\tau . s_\nu)_{\tau\to\nu} \mid (s_{\tau\to\nu}\, t_\tau)_\nu$$

where $c_\tau \in \Sigma_\tau$ is a constant symbol from the (typed) signature $\Sigma := \bigcup_\tau \Sigma_\tau$ and X_τ is a variable. The type of a term is explicitly stated as subscript but may be dropped for legibility reasons if obvious from the context. Terms s_o of type o are *formulas*.

In general, we require Σ to contain a complete logical signature. To that end, we choose Σ to consist at least of the primitive logical connectives for disjunction, negation, and, for each type, equality and universal quantification. Hence, we have $\{\vee_{o\to o\to o}, \neg_{o\to o}, =^\tau_{\tau\to\tau\to o}$ $\Pi^\tau_{(\tau\to o)\to o}\} \subseteq \Sigma$ for all $\tau \in \mathcal{T}$.[1] Optionally, we add choice operators and definite description operators ι for all types. Depending on the logics we are addressing in the following, the concrete set of constants (hence also connectives, quantifiers, etc.) Σ will actually vary. Often, we will only add further constants to the above ones. In all other cases, we will specify Σ explicitly.

[1] The remaining logical connectives can be defined as usual, e.g. conjunction by $\wedge := \lambda s_o.\lambda t_o.\neg(\neg s \vee \neg t)$.

```
thf(1, type, (p: ($i > $i) > $o)).
thf(2, conjecture, (? [F: $i > $i]:
                      (p @ F = p @ (^ [X: $i]: X)))).
```

Figure 1: A small HOL problem in THF representation.

The semantics of HOL is now briefly addressed. A frame $\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ is a collection of non-empty sets $\mathcal{D}_\tau$ such that $\mathcal{D}_o = \{T, F\}$ (for truth and falsehood, respectively) and $\mathcal{D}_{\tau\to\nu} \subseteq {\mathcal{D}_\nu}^{\mathcal{D}_\tau}$ is a collection of functions from $\mathcal{D}_\tau$ to $\mathcal{D}_\nu$. An *interpretation* is a pair $\mathcal{M} = (\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}, \mathcal{I})$ where $\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ is a frame and $\mathcal{I}$ is a function mapping each constant c_τ to some denotation in $\mathcal{D}_\tau$. We assume that the primitive logical connectives are assigned their usual denotation. Given a variable assignment g we can define a valuation $\|.\|^{\mathcal{M},g}$ by

$$\begin{array}{rcl} \|c_\tau\|^{\mathcal{M},g} & = & \mathcal{I}(c_\tau) \\ \|X_\tau\|^{\mathcal{M},g} & = & g(X_\tau) \\ \|s_{\tau\to\nu}\ t_\tau\|^{\mathcal{M},g} & = & \|s_{\tau\to\nu}\|^{\mathcal{M},g}\ \|t_\tau\|^{\mathcal{M},g} \\ \|\lambda X_\tau.\, s_\nu\|^{\mathcal{M},g} & = & \left(f : z \longmapsto \|s\|^{\mathcal{M},g[z/X_\tau]}\right) \in D_{\tau\to\nu} \end{array}$$

where $g[z/X_\tau]$ denotes the variable assignment that maps X_τ to z and every other variable Y_ν to $\sigma(Y_\nu)$, where $Y_\nu \neq X_\tau$.

A formula s_o is called valid, iff $\|s_o\|^{\mathcal{M},g} = T$ for every variable assignment g and every interpretation $\mathcal{M}$. We call $\mathcal{M}$ a *standard model* iff $\mathcal{D}_{\tau\to\nu}$ is the complete set of total functions, i.e. $\mathcal{D}_{\tau\to\nu} = \mathcal{D}_\nu^{\mathcal{D}_\tau}$. As a consequence of Gödel's Incompleteness Theorem [17], HOL with standard semantics is necessarily incomplete. However, if we allow $\mathcal{D}_{\tau\to\nu}$ to be a proper subset of $\mathcal{D}_\nu^{\mathcal{D}_\tau}$ with the constraint that $\|.\|$ remains total, a meaningful notion of completeness can be achieved [19]. We assume this so-called *Henkin semantics* in the following.

A de-facto standard representation of HOL problems for automated theorem provers is given by the THF dialect [35] of the TPTP syntax [34]. This representation syntax is supported by most current HOL ATP, including Satallax [11], LEO-II [4], agsyHOL [20], Isabelle/HOL [22] and many others. A small example problem encoded in THF is displayed in Fig. 1. The circumflex `^` and the `@` denote λ-abstraction and function application, respectively. Types can be stated explicitly (cf. first line of Fig. 1), where `>` denotes the function type constructor $\to$. Most remaining operations are standard TPTP syntax as used in first-order syntax.

We will use the THF dialect as a starting point for the development of specific representations of quantified non-classical logics in the following section.

3 Representation of Non-Classical Logics

In this section, we outline possible conservative extensions to the TPTP THF dialect in order to capture various quantified non-classical logics, to be used as input language of suitable ATP systems. The here discussed logics are modal logics, hybrid logics, conditional logics and free logics, each of them in a higher-order quantified version. Also, we briefly discuss means of representation for many-valued logics.

3.1 Modal Logics

"Modal logic" refers to a family of non-classical logics that are used to express and reason about modal qualities of truth. To that end, the operators $\Box$ and $\Diamond$ are added to the usual

classical logic language and characterized by appropriate rules and axiomatizations. Notions of necessity and possibility are probably the most prominent of such modal concepts represented by the new operators, but many further related systems and interpretations (e.g. focusing on temporal or deontic aspects) exist. Modal logics are not only of strong interest for the interpretation of philosophical arguments, but have also become increasingly important to mathematics and computer science [18].

Syntax and semantics. We now briefly sketch the syntax and semantics of higher-order modal logics (HOML) [21] by augmenting the appropriate definitions of HOL as given in §2. We here assume a multi-modal logic, that is a modal logic consisting of multiple, different box operators $\Box^i$, $i \in I$ (and corresponding diamond operators), for some index set I.

The syntax definition is nearly identical to that of HOL. We merely add the box operators $\Box^i_{o\to o}$ (for all $i \in I$) to the set of constants Σ. Their duals, the diamond operators $\diamond^i$, can be defined by $\diamond^i_{o\to o} := \lambda\Phi_o.\, \neg(\Box^i(\neg\Phi))$.

For the semantics of HOML, we augment the concept of a HOL model with Kripke (possible world) semantics, yielding a HOML model structure $\mathcal{M}$:

$$\mathcal{M} = \left(W, \{R^i\}_{i\in I}, \{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}, \{\mathcal{I}_w\}_{w\in W}\right)$$

where W is a set of worlds, the $R^i \subseteq W \times W$ are accessibility relations between the worlds of W, and each $\mathcal{I}_w$ is an interpretation function (similar to $\mathcal{I}$ of §2) for world w. We assume that connectives are always given the standard interpretation by $\mathcal{I}_w$ in each world $w \in W$.

In a final step, we augment the valuation function $\|.\|^{\mathcal{M},g}$ for HOL models to a valuation function $\|.\|^{\mathcal{M},g,w}$ for HOML models $\mathcal{M}$, a variable assignment g and a world $w \in W$ by

$$\|\Box^i s_o\|^{\mathcal{M},g,w} = T \text{ iff } \text{ for all } v \in W \text{ such that } wR^iv \text{ it holds that } \|s_o\|^{\mathcal{M},g,v} = T$$

The semantics definition stated here are only adequate for *constant domain* semantics in which we assume the domains $\mathcal{D}_\tau$ to be the same for all worlds $w \in W$. However, if we assume *varying domain* semantics (or their restricted forms of cumulative or decreasing domains), we need to further augment the above model. Instead of a single frame $\mathfrak{D} := \{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ we employ a family of frames $\{\mathfrak{D}_w\}_{w\in W}$, one for each world. Additionally, the valuation of universal quantification is appropriately adjusted. We refer to the literature for details (cf. e.g. [15]). Regarding the quantification constants Π^τ, $\tau \in \mathcal{T}$, we might want to allow mixed-semantics quantification statements, i.e. formulas where multiple quantifications are contained, each possibly with different semantics.[2] This can simply be done by adding different quantification constants for the respective different quantification semantics to the signature, yielding $\Sigma = \{\ldots, \Pi^{\tau,co}, \Pi^{\tau,va}, \Pi^{\tau,inc}, \Pi^{\tau,dec}\}$ for each type $\tau \in \mathcal{T}$ for constant, varying, cumulative and decreasing domain quantification semantics, respectively.

We are in a higher-order setting. Hence, bridge rules (e.g. $\Box^i\phi \implies \Box^j\phi$) can simply be postulated as axioms (e.g. $\forall\phi(\Box^i\phi \implies \Box^j\phi)$). Consequently, we can avoid a specific representation for bridge rules below. However, such axioms should possibly be marked specifically so that provers can easily recognize them (similar to TPTP definitions) and apply special techniques where possible.

The above definitions give us modalities with logic K properties. In order to obtain stronger logics, such as KB, KD, $S4$ and $S5$, we e.g. could, analogous to the above bridge rules, postulate respective axioms. However, it seems to us that this approach would be impractical and too verbose. Hence, we include a special syntax for postulating frame conditions to the modalities below.

[2] Mixed uses of constant and varying domain quantifiers occur for example in variants of the ontological argument for the existence of God; cf. Anderson [1], footnote 14.

Representation. We adopt the representation of QMLTP for representing the box and diamond operators, i.e. by writing `#box` for $\Box$ and `#dia` for $\Diamond$. Since we are in a multi-modal setting, we qualify these connectives with an appropriate identifier, called index: `#box(i)` for $\Box^i$ and `#dia(i)` for $\Diamond^i$. The modal operators are then used similar to quantifiers in TPTP: `#box(a): t` represents the formula $\Box^a t_o$. The remaining syntax is standard THF. A short example (where a and b are identifiers from I) is given by:

```
hmf(1, type, (p: $i > $o)).
hmf(2, conjecture, ((#box(a): (! [X: $i]: p @ X))
                    =>
                    (#dia(b): (? [X: $i]: p @ X)))).
```

The `!` quantifier respects some quantification semantics that is chosen by the user. Additionally, we add four new quantifiers `!=`, `!~`, `!+`, `!-` that always denote constant, varying, cumulative and decreasing domain quantification semantics, respectively, regardless of the default setting for `!`. Existential variants are added analogously.

Global parameters. There are several parameters that adjust the exact meaning of modal logic problems. One of this parameters was already mentioned above, namely whether we use constant domain or variants of varying domain semantics. Another important point is the rigidity of constant symbols: Does every symbol denote the same object in every world? The global parameters for the problem input considered here are:

`quantification` Sets the quantification semantics for the `!` and `?` symbols of the language.
Valid values: `constant`, `varying`, `cumulative`, or `decreasing`.
Default value: `constant`.

`constants` Sets the default interpretation constraint for constant symbols, i.e. whether constant symbols have the same denotation in every world (called rigid) or not.
Valid values: `rigid` or `dependent`.
Default value: `rigid`.

`consequence` Specifies the precise meaning of the logical consequence relation $S \models t$ where $S = \{s_1, \ldots, s_n\}$ is a set of formulas. In the `global` case we have: $S \models t$ iff $\forall M, g. \forall w. ((\|s_1\|^{M,g,w} = T \text{ and } \ldots \text{ and } \|s_n\|^{M,g,w} = T)$ implies $\|t\|^{M,g,w} = T)$. In the `local` case we instead have $S \models t$ iff $\forall M, g. ((\forall w. \|s_1\|^{M,g,w} = T \text{ and } \ldots \text{ and } \forall w. \|s_n\|^{M,g,w} = T)$ implies $\forall w. \|t\|^{M,g,w} = T)$.
Valid values: `local` or `global`.
Default value: `local`.

`modalities` Sets which different modalities are defined within the problem. For each indexed modality, the respective index name is given. If not stated, a mono-modal logic is assumed where the default box and diamond operators (i.e. `#box` and `#dia` without name qualification) are used in the problem.
Example: `(a,s5)` defines an indexed modality named `a` with S5 axiomatization.

These parameters need to be included in the problem description using the TPTP process instruction language (TPI)[3] which, amongst other aspects, allows adding meta statements about the problem setting. An exemplary multi-modal setting with cumulative domain semantics, rigid constant interpretation, and a global consequence relation is given by

```
tpi(1, set_logic, modal(['quantification' = 'cumulative',
                         'constants' = 'rigid',
                         'consequence' = 'global'
                         'modalities' = [(a, s5), (b, kb), (c, k)]])).
```

[3] A proposal for the TPI language can be found at http://www.cs.miami.edu/~tptp/TPTP/Proposals/TPILanguage.html.

Here, three different indexed modalities `a`, `b` and `c` are introduced with the given axiomatizations S5, KB and K, respectively. Valid axiomatization schemes for modalities include `k`, `kb`, `k4`, `k5`, `d`, `m`, `b`, `s4` and `s5`. There have been more systems presented in the literature. Hence, this list could/should be appropriately extended.

Per-Symbol Options. As a convenience feature, we allow per-symbol specification of rigidity, allowing some symbols to be rigid and some symbols to be world-dependent. More specifically, all symbols introduced using `type` statements have the default rigidity as stated by the `constants` option of the `set_logic` statement, unless overridden by another `set_logic` statement specifically for that new constant. In the following example, the constant symbol `q` is a rigid symbol while `p` is world-dependent as stated by statement `4`:

```
tpi(1, set_logic, modal(['quantification' = 'cumulative',
                         'constants' = 'rigid',
                         'modalities' = [(a, s5), (b, kb), (c, k)]])).
hmf(2, type, (q: $i)).
hmf(3, type, (p: $i)).
tpi(4, set_logic, hmf(p, 'dependent')).
```

3.2 Hybrid Logics

Hybrid logic [10] is a general term for extensions of ordinary modal logics that introduce a new sort of atomic formulas – the so-called *nominals*. Nominals introduce one convenient feature to modal logic, that is referencing and arguing about worlds. In natural language it is a common construct to refer to a specific point in time or the knowledge of one particular person. Although modal logic is used to model these problem domains, it is not possible to reference to the underlying world structure. Hybrid logic allows to evaluate a formula in a specific world – with the *satisfaction operator* @ – and to bind the current world to a variable – with the *shift operator* $\downarrow$.

Syntax and semantics. Classically, nominals are introduced in propositional and first-order logic by introducing new cases into the syntax BNF and adjusting the models accordingly. Additional to HOML we need to introduce the nominals, the satisfaction operator @, and the *shifter* $\downarrow$ to the BNF over a set NOM.[4]

$$s, t ::= ...|n_o|@(n_o)s_o|\downarrow s_{o\to o}|... \qquad n \in NOM$$

The semantics is build on the same model as ordinary modal logic.

$$\mathcal{M} = \left(W, \{R^i\}_{i\in I}, \{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}, \{\mathcal{I}_w\}_{w\in W}\right)$$

The variable assignment $g = (g^i, g^n)$ is adorned with an extra variable assignment g^n, that maps nominals to worlds. The assignment g^i is the variable assignment from standard higher-order modal logic.

The valuation $\|.\|$ is then augmented for formulas containing nominals, given by

$$\begin{aligned} \|n\|^{\mathcal{M},g,w} &= g^n(n) \equiv w. \qquad n \in NOM \\ \|@n\varphi\|^{\mathcal{M},g,w} &= \|\varphi\|^{\mathcal{M},g,g^n(n)} \\ \|\downarrow\varphi_{o\to o}\|^{\mathcal{M},g,w} &= \|\varphi n\|^{\mathcal{M},g,w} \qquad \text{where } n \text{ is free in } \varphi \\ & \qquad\qquad\qquad\quad \text{and } g^n(n) = w \end{aligned}$$

[4] Higher-Order Hybrid Logics have not yet been researched. They can be conceived as a straight-forward adaptation of propositional and first-order hybrid logic to HOL. Another possibility is to introduce a new type for nominals. This would yield a solution, that resembles the embedding approach [36].

Since hybrid logic is an extension of ordinary modal logic, we can still obtain the stronger logics stated by the frame conditions K B, K D, S4, S5 as usual. On the other hand, nominals allow the formulation of frame conditions, that were not previously expressible in ordinary modal logic. For example, the condition $\Box n \Rightarrow \neg n$ for $n \in NOM$, for instance, corresponds to an irreflexivity condition. As in the last section we allow to name the new frame conditions explicitly.

Representation. We extend the `hmf` syntax for ordinary modal logic described before and add the two new syntax features described above. To distinguish nominals from ordinary Boolean constants, we introduce a new type, called `$nominalType`. In a formula the nominal is grounded to Boolean type, by introducing a new predicate `#nom`. A nominal n has then to be written as `#nom(n)` inside a formula. We propose to adopt the satisfaction and shifter operator with an explicit binding mechanism: `#at(n) : p` for $@n\,p$, and `#shift[n] : p` for $\downarrow(\lambda X\,.\,p)$. Thereby, in both cases, the operator takes as first argument only nominals. The remaining syntax containing modal operators and higher-order features are the above described `hmf` based on the standard THF. A short example (where `n1` is a nominal) is given by:

```
hhf(1, type, p : $i > $o).
hhf(2, type, n1 : $nominalType).
hhf(3, conjecture, (#shift[n] : (#at(n1) :
                  ![X : $i] : (p(X) => #at(n) : p(X))))).
```

Global parameters. Hybrid logic is an extension to ordinary modal logic, with the same parameters. The same options for `quantification, constants`, and `consequence` can be given, as well as giving each symbol a `rigid` or `dependent`. In theory a range of additional frame conditions can be given for the `modalities`, but in practice hybrid logic is used in common frame settings. Hence we suggest to use the exact same (resp. suitably adapted) notation as for ordinary higher-order modal logic.

3.3 Conditional Logics

Conditional logics [24] have many applications including action planning, counter-factual reasoning, default reasoning, deontic reasoning, metaphysical modeling and reasoning about knowledge. A new operator for so-called conditionality, denoted $\longrightarrow$, is added to the basic logical language which is not to be confused with material implication ($\Longrightarrow$).[5] First-order conditional logics have been studied in [14, 16] and extended (to include propositional quantification) and embedded in HOL in [3]. We here consider a higher-order quantified version of conditional logic (HCL).[6]

Syntax and semantics. Terms of HCL are defined as ordinary HOL terms, except that we add a new constant symbol for conditionality to the signature, i.e. $\Sigma = \{\ldots, \longrightarrow_{o\to o\to o}\}$. As for modal logics above, we may add quantifiers for different domain conditions. Monomodal logics are subsumed by HCL since $\Box s_o$ can be introduced as an abbreviation for $\neg s_o \longrightarrow s_o$. Syntactically, HCL can be seen as a generalization of HOML where the index of modality $\longrightarrow$ is a formula of the same language. For instance, in $(s_o \longrightarrow t_o) \longrightarrow u_o$ the subformula $s \longrightarrow t$ is the index of the second occurrence of $\longrightarrow$.

[5] The literature on conditional logics often uses $\Longrightarrow$ for conditionality and $\longrightarrow$ for material implication. Our choice here is pragmatically motivated, since the TPTP already reserves `=>` for material implication.

[6] The extension of quantified conditional logic to full higher-order conditional logic as presented here is ad hoc and straight-forward. Whether there are any particular complications arising from that extension still needs to be inspected.

ID	Axiom	$A \longrightarrow A$
	Condition	$f(w,[A]) \subseteq [A]$
MP	Axiom	$(A \longrightarrow B) \implies (A \implies B)$
	Condition	$w \in [A] \implies w \in f(w,[A])$
CS	Axiom	$(A \wedge B) \implies (A \longrightarrow B)$
	Condition	$w \in [A] \implies f(w,[A]) \subseteq \{w\}$
CEM	Axiom	$(A \longrightarrow B) \vee (A \longrightarrow \neg B)$
	Condition	$\lvert f(w,[A])\rvert \leq 1$
AC	Axiom	$(A \longrightarrow B) \wedge (A \longrightarrow C) \implies (A \wedge C \longrightarrow B)$
	Condition	$f(w,[A]) \subseteq [B] \implies f(w,[A \wedge B]) \subseteq f(w,[A])$
RT	Axiom	$(A \wedge B \longrightarrow C) \implies ((A \longrightarrow B) \implies (A \longrightarrow C))$
	Condition	$f(w,[A]) \subseteq [B] \implies f(w,[A]) \subseteq f(w,[A \wedge B])$
CV	Axiom	$(A \longrightarrow B) \wedge \neg(A \longrightarrow \neg C) \implies (A \wedge C \longrightarrow B)$
	Condition	$(f(w,[A]) \subseteq [B]$ and $f(w,[A]) \cap [C] \neq \emptyset) \implies f(w,[A \wedge C]) \subseteq [B]$
CA	Axiom	$(A \longrightarrow B) \wedge (C \longrightarrow B) \implies (A \vee C \longrightarrow B)$
	Condition	$f(w,[A \vee B]) \subseteq f(w,[A]) \cup f(w,[B])$

Figure 2: Conditional logic axioms and semantic conditions

An adequate semantics is achieved by adapting selection function semantics [31, 12]. We modify the HOL model structure by adding possible worlds (similar to HOML) and a selection function $f : W \times 2^W \mapsto 2^W$, yielding a HCL model structure

$$\mathcal{M} = (W, f, \{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}, \{\mathcal{I}_w\}_{w\in W})$$

where W is a set of worlds, $\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ is a frame and $\{\mathcal{I}_w\}_{w\in W}$ a collection of interpretation functions.

Together with a variable assignment g and a world $w \in W$ we can then refine the valuation function $\|.\|^{M,g,s}$ (only the valuation of conditionality is shown, the remaining cases are straight-forward adaptions of the HOL case):

$$\|s_o \longrightarrow t_o\|^{\mathcal{M},g,w} = T \text{ iff } \|t\|^{\mathcal{M},g,t} \text{ for all } t \in W \text{ s.t. } t \in f(w,[s])$$

where $[s_o] := \{u \mid \|s_o\|^{\mathcal{M},g,u} = T\}$ is the so-called proof set of s.

Like in the case of modal logics, where we distinguish between logics such as K, B, D, S4, S5, there are many different conditional logics, which differ regarding the particular axioms/conditions associated with the conditionality operator $\longrightarrow$. These logics are based on the axioms ID, MP, CS, CEM, AC, RT, CV, CA; see Fig. 2.

Representation. Representation of HCL is straight-forward: The syntax is exactly the same as for ordinary HOL problems. We merely add a new implication `->` which denotes the conditional implication whereas the included implication of THF, `=>`, still denotes material implication (for the sake of consistency). Also, we add the different new quantification symbols `!=`, `!~`, `!+` and `!-` for the respective quantification semantics and denote by `!` the default quantification semantics as chosen by the user. The following example presents a formula that is valid in **MP** but not in **ID**:

```
hcf(1, type, (f : $i > $o)).
hcf(2, type, (g : $i > $o)).
hcf(3, conjecture, (! [X: $i]: ((f @ X) -> (g @ X)) => ((f @ X) => (g @ X)))).
```

Parameters. The global parameters `quantification` and `constants` are the same as for in HOML. We also allow per-symbol rigidity specification as done for HOML. We do not need the specification of modalities here, instead we only need to specify the logic (axiomatization) under consideration. In contrast to modal logic where there exist mostly standardized naming conventions for important logical systems, this is not as evident for conditional logic. However, the axiom names themselves seem standard enough, hence we can describe the logical system by enumerating the included axioms.

This is done by the parameter

`logic` Sets the semantics for the conditional logic under consideration. More precisely, collects a list of axioms that is to be considered. If omitted, a base conditional logic with none additional axioms is assumed.
Valid values: All of the axiom names of Fig. 2.

In the following example the logical axioms **ID**, **MP** and **CEM** (cf. Fig 2) are assumed for the remainder of the conditional logic problem under consideration.

```
tpi(1, set_logic, hcl(['quantification' = 'cumulative',
                       'constants' = 'rigid',
                       'logic' = ['id', 'mp', 'cem']])).
```

Analogous to HOML, the above representations could be extended to support multi-conditional logics, i.e. indexed operators `->`. We do not pursue this further here, since were are not aware yet of applications.

3.4 Free Logics

Classical logic is only mildly suited for handling undefinedness and partiality in an appropriate way. There are two related reasons: (i) terms denote, without exemptions (e.g. for undefined terms), entities in a non-empty domain of "existing" objects D, and (ii) the quantifiers range over this entire set D.

An elegant alternative to remedy these shortcomings is free logic [23, 28], which distinguishes between a raw domain of possibly non-existing objects D and a particular subdomain E of D, containing only the "existing" entities. Free variables range over D and quantified variables only over E. Each term denotes in D, but not necessarily in E. This is the case, for example, for improper definite descriptions which can now be mapped to a distinguished non-existing object, denoted $* \in D$.

Moreover, the domain E may be empty (this special case is called inclusive logic). Unfortunately, no theorem provers have been available so far for free logic. Nevertheless, free logic can be embedded in HOL [8], allowing indirect automation via HOL ATP systems.

Syntax and semantics. The syntax of free higher-order logic is the same as for ordinary HOL. The non-trivial semantics definitions concern universal quantification and definite description (denoted by ι). A model M in this context distinguishes (for all types τ) between a raw domain D_τ and a set $E_\tau \subseteq D_\tau$ of existing objects. A valuation function for these cases can be formulated as

$$\begin{aligned} \|\forall X_\tau.s_o\|^{\mathcal{M},g} &= T \text{ iff } \text{for all } d \in E \text{ holds } \|s_o\|^{\mathcal{M},g[d/X_\tau]} = T \\ \|\iota X_\tau.s_o\|^{\mathcal{M},g} &= \begin{cases} d & \text{if } \{d \in E \mid \|s_o\|^{\mathcal{M},g[d/X_\tau]} = T\} \text{ is unitary} \\ * & \text{otherwise} \end{cases} \end{aligned}$$

Representation. Since the syntax is exactly the same as for HOL, we do not need any special representation for free logic formulas. Nevertheless, we need to specify if the domain E is empty or not.

This is simply stated by the parameter

$\rightarrow$	f	u	t
f	t	t	t
u	u	u	t
t	f	u	t

(a) Strong-Kleene

$\rightarrow$	f	u	t
f	t	u	t
u	u	u	u
t	f	u	t

(b) Weak-Kleene

$\rightarrow^*$	f	u	t
f	t	t	t
u	t	t	t
t	f	f	t

(c) Bocvar

Figure 3: Three different three valued semantics for implication.

`E` Decides whether domain E is allowed to become empty or not.
Valid values: `empty` or `non-empty`. Default value: `empty`

A free logic setting in which the domain E may become empty can be configured using

```
tpi(1, set_logic, free('E' = 'non-empty')).
```

3.5 Many-valued Logics

Classical logics are based on the bivalence principle, that is, the set of truth-values V has the cardinality $|V| = 2$, usually denoted $V = \{T, F\}$ for truth and falsity. Many-valued logics generalize this requirement to more or less arbitrary sets of truth-values, rather referred to as *truth-degrees* in that context. Popular examples of many-valued logics are Gödel logics, Łukasiewicz and fuzzy logics with (non-)denumerable sets of truth-degrees, and, from the field of finitely-many valued logics, Kleene, Bocvar and Dunn/Belnap logic [2]. The latter logics (Kleene, Bocvar) introduce a third value often denoted u for *unknown* and differ in the interpretation in the presence of the unknown value. The Dunn/Belnap logic introduces two additional values n, b – denoting *none* and *both*, respectively.

Many valued logics have applications in linguistics and philosophy (especially non-western philosophy) for arguing about vagueness, and in computer science for analyzing database and information systems.

Syntax and semantics. There is no single unique way for defining the semantics of quantification in many-valued logics. Hence, we are focusing, for the time being, on propositional many-valued logic. The grammar we are considering is formed by

$$a, b ::= t_i \mid c \mid \neg a \mid a \mathcal{C} b.$$

Where the t_i are the truth-values of the logic, $c \in \Sigma$ is a constant symbol and $\mathcal{C}$ is a symbol of the set of binary connectives, containing at least $\{\&, \vee, \wedge, \rightarrow\}$. Depending on the selected logic, additional symbols can be added, and the number of t_i is fixed.

The semantics here is highly dependent on the chosen logic. As usual the semantics of the connectives can be given in a truth-table. In Fig. 3 the semantics of $\rightarrow$ for three different three-valued logics is given as an example.

Representation. Since we are only considering propositional many-valued logic at this point, it does not fit well into the quantified fragment of the TPTP. But since there exist quantified versions of many-valued logics, we base the propositional case on the FOF fragment of the TPTP. This way, the proposal can be extended towards a first-order version.

In addition to FOF, we first introduce a term for truth degrees `#t(i)` for t_i with $i = 1, \ldots, n$, where n is the number of truth constants for the selected logic. To distinguish between weak and strong conjunction, we introduce a new symbol `/\` for the standard *weak*

conjunction $\wedge$. The standard AND-symbol `&` is defined to be the strong conjunction to avoid confusion.

The following is an example for axioms of a many-valued logic, relating $\wedge$ and $\vee$ to the minimal signature of Łukasiewicz or Gödel logics.

```
mvf(1, axiom, (a /\ b) = ((a & (a -> b)))).
mvf(2, axiom, (a | b) = ((a -> b) -> b) /\ ((b -> a) -> a)).
```

Global Parameters. There are two parameters we can provide to adjust the semantics. The first fixes the exact logic. As described, there are many possible many-valued logics, that fix an interpretation for the logical symbols. The second parameter sets the cardinality of the set of truth values. This parameter is only important for the logics with an adjustable amount of truth values. The exact parameters are:

`semantics` Sets the exact logic and fixes the interpretation for the logical symbols.
Valid values: `kleene-weak, kleene-strong, post, lukasiewicz, goedel, bocvar`.

`card` Sets the size of the truth values. Has only an effect on **lukasiewicz** and **goedel**.
Valid values: any natural number

The parameters can be set as in the previous cases in the instruction language TPI. For example, a Łukasiewicz logic with $\{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}$ as truth values can be declared with:

```
tpi(1, set_logic, mvl([‘semantics‘ = ‘lukasiewicz‘,
                       ‘card‘ = 5])).
```

In theory the cardinality cases $\aleph_0$ and $\aleph_1$ were possible, but this would require a different mechanism to name truth values.

3.5.1 MVL based on SIXTEEN

There are several sixteen-valued logics based on a lattice denoted *SIXTEEN*. These logical systems have been developed by Shramko and Wansing as a generalization of the four-valued system of Dunn/Belnap [2] to knowledge bases in computer networks [29] and was subsequently further investigated in various contexts (e.g. [25, 30]). In *SIXTEEN*, the truth-degrees are given by the power set of Belnap's truth values, i.e.

$$V = 2^{\{\mathbf{N},\mathbf{T},\mathbf{F},\mathbf{B}\}} = \{\emptyset, \boldsymbol{N}, \boldsymbol{T}, \boldsymbol{F}, \boldsymbol{B}, \ldots, \{\mathbf{N},\mathbf{T},\mathbf{F},\mathbf{B}\}\}$$

where $\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{F}$ and $\boldsymbol{B}$ are the respective singleton sets containing $\mathbf{N}, \mathbf{T}, \mathbf{F}$ and $\mathbf{B}$. The remaining truth-degrees are named using a combination of the letters $\mathbf{N}, \mathbf{T}, \mathbf{F}$ and $\mathbf{B}$, representing the truth-degree that contains the respective elements when regarded as a set (e.g. $\mathbf{NT}$ for the set $\{\mathbf{N}, \mathbf{T}\}$). This generalization is essentially motivated by the observation that a four-valued system cannot express certain phenomena that arise in knowledge bases in computer networks. Further applications in linguistics and philosophy are discussed in the monograph by Shramko and Wansing [30], to which we refer to for a thorough investigation of *SIXTEEN*, the definitions of logical connectives and their semantics. Briefly speaking, there exists a set of connectives $\vee_\star, \wedge_\star, \neg_\star$ for two distinct logics $\mathcal{L}_\star$ with $\star \in \{t, f\}$ and a logic given by their union, denoted $\mathcal{L}_{tf}$. Additionally, multiple different entailment relations $\models_\star$ for $\star \in \{t, f, tf\}$ can be considered. An embedding of logics based on *SIXTEEN* into HOL for use in ATP systems is sketched in [32].

Representation. The representation of the $\mathcal{L}_\star$ is more involved as we need a non-quantified language with different infix operators denoting the different logical connectives. The following syntax representation could be seen as a generalization and restriction of

FOF, where we add new infix connectives but restrict the problem not to contain first-order ingredients (such as quantifiers). Consequently, we add infix operators `&t`, `|t`, `~t`, `=>t`, `&f`, `|f`, `~f` and `=>f` to the language. The problem statements are then straight-forward, an example is given by

```
sxf(1, axiom, (a |t b)).
sxf(2, axiom, (a |f b)).
sxf(3, conjecture, (a =>t b)).
```

where `a`, `b` are ad-hoc introduced individuals symbols as supported by FOF.

4 Conclusion

In this paper, we discussed means of representing both problems and meta-logical specification for quantified non-classical logics. To that end we adapt and extend TPTP-THF and QMLTP syntax for problems and the TPI language proposal for fixing semantic parameters. We have outlined specialized syntaxes for higher-order modal logic, hybrid logic, conditional logic, and free logic to be used as input languages of ATP systems. Additionally, we sketched ideas for representing many-valued logics. Further logics can easily be added and addressed along the same lines. For example, due to space restrictions we have omitted the inclusion of (quantified) intuitionistic logic [6].

The suggestions in this paper are, at this stage, not meant to be conclusive. Instead, we want to stimulate discussions, e.g. at the ARQNL event, about further requirements and extensions. Moreover, we envision a close collaboration with the QMLTP and TPTP projects.

A concluding, motivating example is displayed in Appendix A where an encoding of Gödel's Ontological argument is given in `hmf` syntax.

Acknowledgments: We thank Harold Boley for his comments and for proofreading this document. We also thank the reviewers for the very valuable feedback they provided.

References

[1] C. A. Anderson. Some emendations of Gödel's ontological proof. *Faith and Philosophy*, 7(3), 1990.

[2] N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logic*, pages 7–37. Reidel Publishing Company, Boston, 1977.

[3] C. Benzmüller. Automating quantified conditional logics in HOL. In F. Rossi, editor, *23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 746–753, Beijing, China, 2013.

[4] C. Benzmüller, L. C. Paulson, N. Sultana, and F. Theiß. The higher-order prover LEO-II. *Journal of Automated Reasoning*, 55(4):389–404, 2015.

[5] C. Benzmüller and B. Woltzenlogel Paleo. The inconsistency in Gödel's ontological argument: A success story for AI in metaphysics. In *IJCAI 2016*, 2016.

[6] Christoph Benzmüller and Lawrence Paulson. Multimodal and intuitionistic logics in simple type theory. *The Logic Journal of the IGPL*, 18(6):881–892, 2010.

[7] Christoph Benzmüller and Lawrence Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.

[8] Christoph Benzmüller and Dana Scott. Automating free logic in Isabelle/HOL. In G.-M. Greuel, T. Koch, P. Paule, and A. Sommese, editors, *Mathematical Software – ICMS 2016, 5th International Congress, Proceedings*, volume 9725 of *LNCS*, Berlin, Germany, 2016. Springer. To appear.

[9] Christoph Benzmüller and Bruno Woltzenlogel Paleo. Higher-order modal logics: Automation and applications. In Adrian Paschke and Wolfgang Faber, editors, *Reasoning Web 2015*, number 9203 in LNCS, pages 32–74, Berlin, Germany, 2015. Springer.

[10] P Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of IGPL*, 8(3):339–365, 2000.

[11] C.E. Brown. Satallax: An automated higher-order prover. In B. Gramlich, D. Miller, and U. Sattler, editors, *Proc. of IJCAR 2012*, volume 7364 of *LNAI*, pages 111 – 117. Springer, 2012.

[12] B.F. Chellas. Basic conditional logic. *Journal of Philosophical Logic*, 4(2):133–153, 1975.

[13] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[14] J.P. Delgrande. On first-order conditional logics. *Artificial Intelligence*, 105(1-2):105–137, 1998.

[15] M. Fitting and R.L. Mendelsohn. *First-Order Modal Logic*. Synthese Library Studies in Epistemology Logic, Methodology, and Philosophy of Science Volume 277. Springer, 1998.

[16] N. Friedman, J.Y. Halpern, and D. Koller. First-order conditional logic for default reasoning revisited. *ACM Transactions on Computational Logic*, 1(2):175–207, 2000.

[17] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[18] R. Goldblatt. Mathematical modal logic: a view of its evolution. *Journal of Applied Logic*, 1(5):309–392, 2003.

[19] L. Henkin. Completeness in the theory of types. *Journal Symbolic Logic*, 15(2):81–91, 1950.

[20] F. Lindblad. agsyHol website. https://github.com/frelindb/agsyHOL, 2012.

[21] Reinhard Muskens. Higher order modal logic. *Handbook of modal logic*, 3, 2007.

[22] T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.

[23] J. Nolt. Free logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2014 edition, 2014.

[24] D. Nute. *Topics in conditional logic*. Reidel, Dordrecht, 1980.

[25] S. P. Odintsov. On Axiomatizing Shramko-Wansing's Logic. *Studia Logica*, 91(3):407–428, 2009.

[26] J. Otten. MleanCoP: A Connection Prover for First-Order Modal Logic. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning: 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, pages 269–276, Cham, 2014. Springer.

[27] T. Raths and J. Otten. The QMLTP Problem Library for First-Order Modal Logics. In B. Gramlich, D. Miller, and U. Sattler, editors, *IJCAR 2012*, volume 7364 of *LNCS*, pages 454–461. Springer, 2012.

[28] D. Scott. Existence and description in formal logic. In R. Schoenman, editor, *Bertrand Russell: Philosopher of the Century*, pages 181–200. George Allen & Unwin, London, 1967.

[29] Y. Shramko and H. Wansing. Some useful 16-valued logics: How a computer network

should think. *Journal of Philosophical Logic*, 34(2):pp. 121–153, 2005.

[30] Y. Shramko and H. Wansing. *Truth and Falsehood: An Inquiry into Generalized Logical Values*. Trends in Logic. Springer Netherlands, 2011.

[31] R.C. Stalnaker. A theory of conditionals. In *Studies in Logical Theory*, pages 98–112. Blackwell, 1968.

[32] A. Steen and C. Benzmüller. Sweet SIXTEEN: Automation via Embedding into Classical Higher-Order Logic. In *7th International Conference Non-Classical Logic – Theory and Applications, Toruń, Poland*, 2015.

[33] A. Steen, M. Wisniewski, and C. Benzmüller. Agent-based HOL reasoning. In G.-M. Greuel, T. Koch, P. Paule, and A. Sommese, editors, *The 5th International Congress on Mathematical Software (ICMS 2016)*, volume 9725 of *LNCS*, Berlin, Germany, 2016. Springer. To appear.

[34] G. Sutcliffe. The TPTP problem library and associated infrastructure. *J. Autom. Reasoning*, 43(4):337–362, 2009.

[35] 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.

[36] M. Wisniewski and A. Steen. Embedding of Quantified Higher-Order Nominal Modal Logic into Classical Higher-Order Logic. In C. Benzmüller and J. Otten, editors, *1st International Workshop on Automated Reasoning in Quantified Non-Classical Logics (ARQNL 2014) Vienna, Austria, Proceedings*, volume 33 of *EasyChair Proceedings in Computing*, pages 59–64. EasyChair, 2014.

A Example: Gödel's ontological argument

The following example is an encoding of Gödel's ontological argument in `hmf` syntax as described in this work (cf. §3.1). More precisely, we present here Dana Scott's variant as a one step proof in which intermediate argumentation steps are omitted. HOL provers such as Leo-II are capable of automating this example in a few seconds (however, not yet for the syntax representation below). Details about the ontological argument, its formalization and its automation can be found in the literature (e.g. in [5]).

```
tpi(1,set_logic,modal(['quantification' = 'varying',
                       'constants' = 'rigid',
                       'consequence' = 'global'
                       'modalities' = [(a, s5)]])).

hmf(positive_const,type,(p: ($i>$o)>$o)).
hmf(A1,axiom,(![Phi: $i>$o]:
    ((p @ (^[X:$i]: ~(Phi @ X)) <=> ~(P @ Phi))))).
hmf(A2,axiom,(![Phi: $i>$o,Psi: $i>$o]:
    (((p @ Phi)
      & #box(a): (![X:$i]: ((Phi @ X) => (Psi @ X))))
     => (p @ Psi)))).
hmf(god_const,type,(g: $i>$o)).
hmf(god,definition,(g =
    (^[X:$i]: (![Phi: $i>$o]: ((p @ Phi) => (Phi @ X)))))).
hmf(A3,axiom,(p @ g)).
hmf(A4,axiom,(![Phi: $i>$o]: ((p @ Phi) => #box(a): (p @ Phi)))).
hmf(essence_const,type,(ess: ($i>$o)>$i>$o)).
hmf(essence,definition,(ess =
    (^[Phi: $i>$o,X: $i]:
     ((Phi @ X)
      & (![Psi: $i>$o]:
         ((Psi @ X)
          => #box(a): (![Y:$i]: ((Phi @ Y) => (Psi @ Y)))))))))).
hmf(necessary_existence_const,type,(ne: $i>$o)).
hmf(necessary_existence,definition,(ne =
    (^[X:i]: (![Phi: $i>$o]: ((ess @ Phi @ X)
                              => #box(a): (?[Y:$i]: (Phi @ Y)))))))).
hmf(A5,axiom,(p @ ne)).
hmf(T3,conjecture,(#box(a): (?[X:$i]: (g @ X)))).
```