Translating Higher-Order Modal Logic from RuleML to TPTP

Harold Boley[1] Christoph Benzmüller[2] Meng Luan[3] Zhendong Sha[1]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
{harold.boley, z.sha}[AT]unb.ca
[2] Institute of Computer Science, Freie Universität Berlin, Germany
c.benzmueller[AT]fu-berlin.de
[3] RuleML Inc, Fredericton, Canada
edmonluan[AT]gmail.com

Abstract. This paper discusses extensions of RuleML and TPTP for higher-order logic, modal logic, and higher-order modal logic. For these extensions, an extended XSLT 2.0-based translator is presented which maps from RuleML/XML to TPTP. With this implemented HOML RuleM2TPTP tool, TPTP can benefit from RuleML interoperation and much of Deliberation RuleML can execute on TPTP-aware provers.

1 Introduction

Higher-order logic [1] is gaining momentum partly since functional languages (e.g., using lambda expressions) have become mainstream with Java 8 [2]. Modal logic [3] is getting traction partly through business rules (e.g., about obligations and permissions) in industry standards such as SBVR [4]. We use the term "higher-order modal logic" (HOML) to denote these logics and their combination.

RuleML [5–8][4] is a system of families of languages at the center of a knowledge-interoperation hub. "Thousands of Problems for Theorem Provers" (TPTP) [9][5] is a widely used syntax and library for Automated Theorem Proving (ATP) test/benchmark problems.

Driven, e.g., by the above practical considerations, both the RuleML and TPTP communities have felt an increasing need to expand their systems for HOML. In an effort to join forces, this paper demonstrates an XSLT 2.0-based translator from HOML RuleML/XML to HOML TPTP. Because of the tree-structured XML source, this is considered easier than an inverse translator.

The translator of HOML from RuleML to TPTP extends an earlier translator from FOL (first-order logic) RuleML to TPTP FOF (first-order form): RuleML2TPTP[6] is an XSLT 2.0-based translator from Deliberation RuleML/XML to TPTP. Originally implemented for Datalog$^+$ RuleML, it was later extended to Hornlog$^+$ RuleML, and then to all of FOL RuleML (with Equality).

[4] http://ruleml.org.
[5] http://www.cs.miami.edu/~tptp/.
[6] http://wiki.ruleml.org/index.php/TPTP_RuleML.

Like this earlier FOL RuleML2TPTP, the newly implemented HOML RuleML2TPTP performs recursive case analysis to linearize XML trees to TPTP texts. The generated TPTP syntax can be validated and executed with TPTP-aware higher-order ATP systems such as Leo-II [10], Satallax [11], Isabelle [12] and the new Leo-III [13]. The latter system is currently extended to also support a recently proposed TPTP syntax extension for HOML [14].

The current paper is organized as follows. Section 2 presents the evolving HOML TPTP. Section 3 proposes the new HOML RuleML. Section 4 describes the RuleML-to-TPTP translation method. Section 5 illustrates the translation with examples. Section 6 explains the XSLT-based translator. Finally, Section 7 gives conclusions.

2 HOML TPTP

HOML extends higher-order classical logic with a (multi-)modal logic using a countable set of (parameterized) dual pairs of abstract modal 'box'/'diamond' operators, where the first two pairs might be instantiated to, e.g., 'necessary'/'possible' (alethically) and 'obligatory'/'permissible' (deontically).

For higher-order classical logic the TPTP THF languages have been proposed [15]. THF stands for "typed higher-order form" and it refers to a family of syntax formats for higher-order logic (HOL). So far, only the fully developed TH0 format, for simple type theory, is in practical use. However, in an ongoing effort, a conservative extension of TH0, called TH1, has emerged [16]. In contrast to TH0, TH1 supports rank-1 polymorphism.

To capture typed higher-order modal logics, another extension of TH0, called THM, is currently under development [14], using the language identifier hmf (higher-order modal form). The work presented here is aiming at capturing first TH0 and subsequently THM (or its possible merge with TH1).

In TH0, which is a concrete syntax for HOL, `$i` and `$o` represent the HOL base types i (individuals) and o (Booleans). `$i>$o` encodes a function (predicate) type. Predicate application, as in $A(X, W)$, is encoded as in `((A@X)@W)` or simply in `(A@X@W)`, i.e. function/predicate application is represented by `@`; universal quantification and λ-abstraction, as in $\lambda A_{i\to o} \forall W_i A(W)$, are encoded as in `^[A:$i>$o]:![W:$i]:(A@W)`.

3 HOML RuleML

HOML RuleML is an advancement plus combination of Functional RuleML [17][7], for its HOL RuleML subset, and Modal RuleML [5][8], for its Modal Logic (ML) RuleML subset. HOL RuleML is adapted from Functional RuleML, with the type level added. ML RuleML is inspired by Modal RuleML (linked above) and

[7] `http://ruleml.org/fun`.
[8] `http://ruleml.org/1.0/modal/modal.html`.

by TPTP THM [14]. RuleML 1.02 also provides a semantic profile [18][9] for the (SBVR-)practical special case of first-order deontic-alethic logic [19, 20].

Since HOL RuleML considers predicates (relations) as characteristic (boolean-valued) functions, it applies both of these via RuleML's `Uniterm`s, generalizing `Atom`s and `Expr`essions:[10] `Const`ants are employed as the `op`erators of `Uniterm`s, generalizing `Rel`ations of `Atom`s and `Fun`ctions of `Expr`essions (as well as `Ind`ividuals).[11] While `formula`s point to, e.g., `Atom`s, the RIF-FLD-like `termula`s [21] point to, e.g., `Uniterm`s.

For the type level, HOL RuleML proposes a `Tbase` ("Type `base`") element whose PCDATA currently correspond to `$i` and `$o` (without the "$" prefix). On top of this, two further RuleML/XML elements are proposed: `Entitype` associates an `entity` (e.g., a `Var`iable) with a `type` (e.g., a `Tbase` or a `Tmap`); `Tmap` ("Type `map`") combines – into a function type – a `domain` type and a `codomain` type, both of which may again be `Tmap` types etc., down to any level of nesting.

Since ML RuleML provides a countable set pairing 'necessary'-like parameterized 'box' operators with dual 'possible'-like parameterized 'diamond' operators, it is more abstract than Modal RuleML 1.0. Also, ML RuleML is more expressive since, instead of an `Atom` element with a `modal`-attributed `Rel`ation and a modally embedded atom, ML RuleML uses a new `Modal` element with a `variety` edge leading to `Box` or `Dia`mond elements and a `proposition` edge leading to a modally embedded arbitrary formula.

4 Translation Method

To avoid extending FOL RuleML2TPTP's preprocessing to HOML RuleML/XML using a (yet to be extended) RuleML normalizer, a fully striped normal form[12] is assumed as the input for translation to TPTP output. This form gives XML trees an object-oriented flavor by alternating (upper-cased <Node>) object elements with (lower-cased <edge>) property subelements. Like for earlier subfamilies of Deliberation RuleML, the HOML RuleML normalizer will need to, e.g., reconstruct missing <edge> tags, making all omitted information explicit.

The translation from this normalized XML syntax to the TPTP syntax is specified by mapping tables defining recursive translation functions τ_{lang}below: Each row containing *xml* in the first column and *tptp* in the second column indicates a translation case $\tau_{lang}(xml) = tptp$, where – for most of the key language constructs (beyond FOL) considered here – *tptp* is composed from the results of recursive applications of $\tau_{lang}(xml_i)$ to parts xml_i of *xml*.

[9] `http://ruleml.org/1.02/profiles/fodal`.

[10] `http://wiki.ruleml.org/index.php/Glossary_of_Deliberation_RuleML_1.02#.3CUniterm.3E`.

[11] `http://wiki.ruleml.org/index.php/Glossary_of_Deliberation_RuleML_1.02#.3CConst.3E`.

[12] `http://wiki.ruleml.org/index.php/Specification_of_Deliberation_RuleML_1.02#XSLT-Based_Normalizer`.

In Table 1, the mapping of the key higher-order language constructs with equality (which was already implemented in FOL RuleML2TPTP) and types is given as a translation function τ_{HO}.

Table 1. Mapping from HOL RuleML/XML to HOL TPTP

HOL RuleML/XML: *xml*	HOL TPTP: $\tau_{HO}(xml)$ = *tptp*
<Equal> <left>t_1</left> <right>t_2</right> </Equal>	($\tau_{HO}(t_1)$ = $\tau_{HO}(t_2)$)
<Uniterm> <op>t</op> <arg>t_1</arg> ... <arg>t_n</arg> </Uniterm>	(τ_{HO}(t) @ $\tau_{HO}(t_1)$ @ ... @ $\tau_{HO}(t_n)$)
<Lambda> <declare>t_1</declare> ... <declare>t_n</declare> <termula>t</termula> </Lambda>	(^ [$\tau_{HO}(t_1)$, ... ,$\tau_{HO}(t_n)$] : τ_{HO}(t))
<Entitype> <entity>t_1</entity> <type>t_2</type> </Entitype>	$\tau_{HO}(t_1)$:$\tau_{HO}(t_2)$
<Tmap> <domain>t_1</domain> <codomain>t_2</codomain> </Tmap>	($\tau_{HO}(t_1)$ > $\tau_{HO}(t_2)$)
<Tbase>i</Tbase>	$i
<Tbase>o</Tbase>	$o

In Table 2, the mapping of the key modal language constructs is given as a translation function τ_M.

The combination of the τ_{HO} and τ_M mappings results in the HOML mapping.

5 Translation Examples

The following demo examples of RuleML-to-TPTP translation pairs are available from a GitHub test directory.[13]

[13] https://github.com/RuleML/RuleML2TPTP/blob/homl/src/test/resources/test/translator/HOMLRuleML2TPTP/.

Table 2. Mapping from ML RuleML/XML to ML TPTP

ML RuleML/XML: *xml*	ML TPTP: $\tau_M(xml) = tptp$

`<Modal>`
`  <variety>`t_1`</variety>`
`  <proposition>`t_2`</proposition>`
`</Modal>`
	$(\tau_M(t_1) : \tau_M(t_2))$

`<Box>`
`  <arg>t</arg>`
`</Box>`
	`#box(`τ_M`(t))`

`<Dia>`
`  <arg>t</arg>`
`</Dia>`
	`#dia(`τ_M`(t))`

The initial two examples take HOL TPTP from [15] and give the corresponding normalized HOL RuleML/XML.[14] However, since the TPTP also constitutes the output of our translator, it is depicted after the RuleML/XML as its input.

The first example shows the RuleML-TPTP translation pair for serializing $union = \lambda X_{\iota\to o}, Y_{\iota\to o}, U_\iota((X\ U) \vee (Y\ U))$.

```
<Equal>
  <left><Const>union</Const></left>
  <right>
    <Lambda>
      <declare>
        <Entitype>
          <entity>
            <Var>X</Var>
          </entity>
          <type>
            <Tmap>
              <domain><Tbase>i</Tbase></domain>
              <codomain><Tbase>o</Tbase></codomain>
            </Tmap>
          </type>
        </Entitype>
      </declare>
      <declare>
        <Entitype>
          <entity>
            <Var>Y</Var>
          </entity>
          <type>
            <Tmap>
              <domain><Tbase>i</Tbase></domain>
              <codomain><Tbase>o</Tbase></codomain>
```

[14] Since the TPTP language identifier (`thf` or `hmf`) and name of a TPTP formula are currently not specified on the RuleML/XML level, they are not yet deployed in the implemented translator.

```
            </Tmap>
          </type>
        </Entitype>
      </declare>
      <declare>
        <Entitype>
          <entity>
            <Var>U</Var>
          </entity>
          <type>
            <Tbase>i</Tbase>
          </type>
        </Entitype>
      </declare>
      <termula>
        <Or>
          <termula>
            <Uniterm>
              <op><Var>X</Var></op>
              <arg><Var>U</Var></arg>
            </Uniterm>
          </termula>
          <termula>
            <Uniterm>
              <op><Var>Y</Var></op>
              <arg><Var>U</Var></arg>
            </Uniterm>
          </termula>
        </Or>
      </termula>
    </Lambda>
  </right>
</Equal>
```

```
thf(union,definition,
    ( union
    = ( ^ [X:($i > $o),Y:($i > $o),U:$i] :
          ( ( X @ U )
          | ( Y @ U ) ) ) )).
```

The second example shows the RuleML-TPTP translation pair serializing $\forall A_{\iota\to o}\forall B_{\iota\to o}\forall C_{\iota\to o}((A \cup (B \cap C)) = ((A \cup B) \cap (A \cup C)))$.

Here, binary (generally, n-ary) function applications are used instead of Currying into nested unary applications (although these are possible too), since both RuleML's `Uniterm`s and TPTP's `@` infixes admit n-ary applications.

```
<Forall>
  <declare>
    <Entitype>
      <entity>
```

```
      <Var>A</Var>
    </entity>
    <type>
      <Tmap>
        <domain><Tbase>i</Tbase></domain>
        <codomain><Tbase>o</Tbase></codomain>
      </Tmap>
    </type>
  </Entitype>
</declare>
<declare>
  <Entitype>
    <entity>
      <Var>B</Var>
    </entity>
    <type>
      <Tmap>
        <domain><Tbase>i</Tbase></domain>
        <codomain><Tbase>o</Tbase></codomain>
      </Tmap>
    </type>
  </Entitype>
</declare>
<declare>
  <Entitype>
    <entity>
      <Var>C</Var>
    </entity>
    <type>
      <Tmap>
        <domain><Tbase>i</Tbase></domain>
        <codomain><Tbase>o</Tbase></codomain>
      </Tmap>
    </type>
  </Entitype>
</declare>
<termula>
  <Equal>
    <left>
      <Uniterm>
        <op><Const>union</Const></op>
        <arg><Var>A</Var></arg>
        <arg>
          <Uniterm>
            <op><Const>intersection</Const></op>
            <arg><Var>B</Var></arg>
            <arg><Var>C</Var></arg>
          </Uniterm>
        </arg>
      </Uniterm>
```

```
      </left>
      <right>
        <Uniterm>
          <op><Const>intersection</Const></op>
          <arg>
            <Uniterm>
              <op><Const>union</Const></op>
              <arg><Var>A</Var></arg>
              <arg><Var>B</Var></arg>
            </Uniterm>
          </arg>
          <arg>
            <Uniterm>
              <op><Const>union</Const></op>
              <arg><Var>A</Var></arg>
              <arg><Var>C</Var></arg>
            </Uniterm>
          </arg>
        </Uniterm>
      </right>
    </Equal>
  </termula>
</Forall>
```

```
thf(union_distributes_over_intersection,conjecture,
    ( ! [A:($i > $o),B:($i > $o),C:($i > $o)] :
        ( ( union @ A @ ( intersection @ B @ C ) )
        = ( intersection @ ( union @ A @ B ) @ ( union @ A @ C ) ) ) )).
```

The third example of a RuleML-TPTP pair illustrates the normalized ML RuleML/XML for a mapping-target ML TPTP formula that is specialized from [14] to the syntax of first-order modal logic, leading to what could be dubbed a first-order modal form (fmf[15]).

```
<Implies>
  <if>
    <Modal>
      <variety>
        <Box><arg><Const>a</Const></arg></Box>
      </variety>
      <proposition>
        <Forall>
          <declare><Var>X</Var></declare>
          <formula>
            <Atom>
              <op><Rel>p</Rel></op>
              <arg><Var>X</Var></arg>
            </Atom>
```

[15] We use fmf for didactic purposes, as a specialization of the hmf language identifier.

```
          </formula>
        </Forall>
      </proposition>
    </Modal>
  </if>
  <then>
    <Modal>
      <variety>
        <Dia><arg><Const>b</Const></arg></Dia>
      </variety>
      <proposition>
        <Exists>
          <declare><Var>X</Var></declare>
          <formula>
            <Atom>
              <op><Rel>p</Rel></op>
              <arg><Var>X</Var></arg>
            </Atom>
          </formula>
        </Exists>
      </proposition>
    </Modal>
  </then>
</Implies>

fmf(1, conjecture,
    (  (#box(a): ( ! [X]: p(X) ))
    => (#dia(b): ( ? [X]: p(X) ))  )).
```

The fourth RuleML-TPTP-pairing example, extending the third, illustrates the syntax of normalized HOML RuleML/XML for the syntax of a HOML TPTP formula as the mapping target taken unchanged from [14].

```
<Implies>
  <if>
    <Modal>
      <variety>
        <Box><arg><Const>a</Const></arg></Box>
      </variety>
      <proposition>
        <Forall>
          <declare>
            <Entitype>
              <entity><Var>X</Var></entity>
              <type><Tbase>i</Tbase></type>
            </Entitype>
          </declare>
          <termula>
            <Uniterm>
              <op><Rel>p</Rel></op>
```

```
              <arg><Var>X</Var></arg>
            </Uniterm>
          </termula>
        </Forall>
      </proposition>
    </Modal>
  </if>
  <then>
    <Modal>
      <variety>
        <Dia><arg><Const>b</Const></arg></Dia>
      </variety>
      <proposition>
        <Exists>
          <declare>
            <Entitype>
              <entity><Var>X</Var></entity>
              <type><Tbase>i</Tbase></type>
            </Entitype>
          </declare>
          <termula>
            <Uniterm>
              <op><Rel>p</Rel></op>
              <arg><Var>X</Var></arg>
            </Uniterm>
          </termula>
        </Exists>
      </proposition>
    </Modal>
  </then>
</Implies>

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

Because of the universal and existential quantification over individuals and because of the different `box`/`dia` arguments `a` vs. `b`, the above example is a first-order multi-modal logic conjecture. Since propositional and first-order (multi-) modal logics are fragments of higher-order (multi-)modal logic, any TPTP-aware prover or model finder for HOML will be applicable to it to analyze its validity (e.g., it is countersatisfiable in a multi-modal version of logic K, while it would be valid after replacing the `dia` argument `b` with `a` in modal logic S4).

6 XSLT Translator

Our translator from the new HOML RuleML to the emerging TPTP for HOML is realized as an extension of the earlier RuleML2TPTP translator from FOL

RuleML to TPTP FOF. Furthermore, as is shown in the second column of Tables 1 and 2 in Section 4, the TPTP constructs are generated as a combination of the symbols of the emerging TPTP for HOML with the results of calling the earlier FOL translation function.

The new table-defined translation functions τ_{HO} and τ_M are implemented in XSLT 2.0. Each row of the tables leads to a corresponding XSLT template, which describes in detail the mapping from HOML RuleML to TPTP.

Note that the namespace prefix "`r:`" for elements is defined at the beginning of the XSLT file, expanding to the RuleML namespace "http://ruleml.org/spec".

The extended translator is available as a GitHub source file.[16]

6.1 Translating Higher-Order Constructs

The HOL translator defines seven XSLT templates to map the XML elements specified for the translation function τ_{HO} as the seven rows of Table 1.

For example, the template for the `Entitype` element is defined as follows:

```
<xsl:template match="r:Entitype">
    <xsl:apply-templates select="r:entity"/>
    <xsl:text>:</xsl:text>
    <xsl:apply-templates select="r:type"/>
</xsl:template>
```

This XSLT definition maps the RuleML/XML `Entitype` prefix notation to a TPTP "`:`" infix notation. The `apply-templates` calls on both sides of the generated "`:`" `text` realize the τ_{HO} recursions in Table 1.

6.2 Translating Modal Constructs

The ML translator defines three XSLT templates to map the XML elements specified for the translation function τ_M as the three rows of Table 2.

For example, the template for the `Modal` element is defined as follows:

```
<xsl:template match="r:Modal">
    <xsl:text>(</xsl:text>
    <xsl:apply-templates select="r:variety"/>
    <xsl:text>: (</xsl:text>
    <xsl:apply-templates select="r:proposition"/>
    <xsl:text>))</xsl:text>
</xsl:template>
```

This XSLT definition maps the RuleML/XML `Modal` prefix notation to a TPTP "`:`" infix notation. The `apply-templates` calls on both sides of the generated "`: (`" `text` realize the τ_M recursions in Table 2.

[16] `https://github.com/RuleML/RuleML2TPTP/blob/homl/src/main/resources/xslt/ruleml2tptp.xslt`.

7 Conclusions

This joint effort continues work that was done separately in the RuleML and TPTP communities: Extensions of both systems for (typed) higher-order logic, modal logic, and combined higher-order modal logic. RuleML's new HOML RuleML/XML, mapping to the evolving HOML TPTP, already provided feedback[17] to TPTP. The resulting XSLT-based translator, HOML RuleML2TPTP, characterizes the proposed HOL and HOML RuleML/XML extension and allows experimenting with HOL and HOML RuleML/XML examples executed by TPTP-aware ATPs for HOL and HOML such as Leo-III.

Future work includes: Mapping RuleML performatives to TPTP roles; defining HOML RuleML with the XML-schema language Relax NG; identifying the language (e.g., as anchors[18] corresponding to TPTP's `thf` or `hmf`) in RuleML/XML instances (e.g., via `schemaLocation`); adapting the current translator for mapping this language identifier to TPTP along with reusing the FOL RuleML2TPTP name generator for HOML RuleML2TPTP's TPTP formulas; implementing an inverse translator, HOML TPTP2RuleML; detailing the correspondences between the earlier Functional RuleML and Modal RuleML and the new HOL RuleML and ML RuleML; as well as introducing RuleML signature declarations (e.g., starting with those in POSL [22] and Reaction RuleML[19]) to be mapped to TPTP THF signatures (e.g., to declare the type of a constant symbol[20]). Other future work could add further translation targets, e.g. mapping to the existing QMLTP syntax format for first-order modal logic [23]; this would enable the experimentation with further ATPs [24].

8 Acknowledgements

We thank Gen Zou for his helpful suggestions, which led to improvements of the proposed HOML RuleML/XML and of this paper. Likewise, we thank the reviewers for their constructive comments as well as the chairs of the 10th International Rule Challenge for organizing this event. The first two authors are grateful to their host, Edward Zalta, at the Center for the Study of Language and Information, Stanford University, for providing an inspiring environment for starting this work. The first author thanks NSERC for its support through Discovery Grants. The work of the second author has been supported by the German Research Foundation DFG under reference number BE2501/9-2.

[17] About parenthesizing TPTP type-mapping formulas, corresponding to RuleML `Tmaps`, and considering possible future replacement of some kinds of colon infixes with alternative infix characters.

[18] `http://deliberation.ruleml.org/1.02/relaxng/#anchor`.

[19] `http://wiki.ruleml.org/index.php/Glossary_of_Reaction_RuleML_1.02#.3Csignature.3E`.

[20] `http://www.cs.miami.edu/~tptp/TPTP/TR/TPTPTR.shtml#THF%20formulae`.

References

1. Farmer, W.M.: The seven virtues of simple type theory. Journal of Applied Logic **6**(3) (2008) 267–286, http://www.sciencedirect.com/science/article/pii/S157086830700081X.
2. Horstmann, C.: Java SE8 for the Really Impatient: A Short Course on the Basics. Java Series. Pearson Education (2014)
3. Garson, J.: Modal Logic. In Zalta, E.N., ed.: The Stanford Encyclopedia of Philosophy. Spring 2016, http://plato.stanford.edu/archives/spr2016/entries/logic-modal/ edn. (2016)
4. de Jesus, J.S., de Melo, A.C.V.: Business Rules: From SBVR to Information Systems. In Fournier, F., Mendling, J., eds.: Business Process Management Workshops. Volume 202 of Lecture Notes in Business Information Processing., Springer (2014) 489–503
5. Boley, H.: The RuleML Family of Web Rule Languages. In Alferes, J.J., Bailey, J., May, W., Schwertel, U., eds.: PPSWR. Volume 4187 of Lecture Notes in Computer Science., Springer (2006) 1–17
6. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Proc. 4th International Web Rule Symposium: Research Based and Industry Focused (RuleML-2010), Washington, DC, USA, October 2010. Lecture Notes in Computer Science, Springer (2010)
7. Athan, T., Boley, H., Paschke, A.: RuleML 1.02: Deliberation, Reaction, and Consumer Families. In: Proceedings of the RuleML2015 Challenge, at the 9th International Symposium on Rules, CEUR (August 2015)
8. Boley, H.: The RuleML Knowledge-Interoperation Hub. In: Proc. 10th International Web Rule Symposium (RuleML 2016), Stony Brook, New York, USA. Volume 9718 of Lecture Notes in Computer Science., Springer (July 2016)
9. Sutcliffe, G.: The TPTP problem library and associated infrastructure. J. Autom. Reasoning **43**(4) (2009) 337–362
10. Benzmüller, C., Paulson, L.C., Sultana, N., Theiß, F.: The higher-order prover LEO-II. Journal of Automated Reasoning **55**(4) (2015) 389–404
11. Brown, C.: Satallax: An automated higher-order prover. In: IJCAR 2012. Number 7364 in LNAI, Springer (2012) 111 – 117
12. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Number 2283 in LNCS. Springer (2002)
13. Steen, A., Wisniewski, M., Benzmüller, C.: Agent-based HOL reasoning. In Greuel, G.M., Koch, T., Paule, P., Sommese, A., eds.: Mathematical Software – ICMS 2016, 5th International Congress, Proceedings. Volume 9725 of LNCS., Berlin, Germany, Springer (2016) To appear.
14. Wisniewski, M., Steen, A., Benzmüller, C.: TPTP and beyond: Representation of quantified non-classical logics. In: ARQNL 2016. Automated Reasoning in Quantified Non-Classical Logics. (2016) To appear.
15. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. Journal of Formalized Reasoning **3**(1) (2010) 1–27
16. Kaliszyk, C., Sutcliffe, G., , Rabe, F.: Th1: The tptp typed higher-order form with rank-1 polymorphism. In: PAAR 2016. (2016) To appear.
17. Boley, H.: Functional RuleML: From Horn Logic with Equality to Lambda Calculus. UPGRADE **VI**(6) (2005)
18. Paschke, A.: RuleML - Semantic Profile for the First Order Deontic Alethic Logic RuleML Inc., Canada, http://reaction.ruleml.org/1.02/profiles/FirstOrderDeonticAlethicLogicProfile.pdf.

19. Marinos, A., Krause, P.J.: An SBVR Framework for RESTful Web Applications. In Governatori, G., Hall, J., Paschke, A., eds.: Rule Interchange and Applications, International Symposium, RuleML 2009, Las Vegas, Nevada, USA, November 5-7, 2009. Proceedings. Volume 5858 of Lecture Notes in Computer Science., Springer (2009) 144–158
20. Solomakhin, D., Franconi, E., Mosca, A.: Logic-based Reasoning Support for SBVR. Fundam. Inform **124**(4) (2013) 543–560
21. Boley, H., Kifer, M.: RIF Framework for Logic Dialects (June 2010) W3C Recommendation, `http://www.w3.org/TR/rif-fld`.
22. Boley, H.: Integrating Positional and Slotted Knowledge on the Semantic Web. Journal of Emerging Technologies in Web Intelligence **4**(2) (November 2010) 343–353
23. Raths, T., Otten, J.: The QMLTP Problem Library for First-Order Modal Logics. In: Automated Reasoning: 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2012) 454–461
24. Benzmüller, C., Otten, J., Raths, T.: Implementing and evaluating provers for first-order modal logics. In Raedt, L.D., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P., eds.: ECAI 2012. Volume 242 of Frontiers in Artificial Intelligence and Applications., Montpellier, France, IOS Press (2012) 163–168