Progress in Automating Higher-Order Ontology Reasoning*

Christoph Benzmüller and Adam Pease

Articulate Software, CA, USA {cbenzmueller,apease}@articulatesoftware.com

Abstract

We report on the application of higher-order automated theorem proving in ontology reasoning. Concretely, we have integrated the Sigma knowledge engineering environment and the Suggested Upper-Level Ontology (SUMO) with the higher-order theorem prover LEO-II. The basis for this integration is a translation from SUMO's SUO-KIF representations into the new typed higher-order form representation language TPTP THF. We illustrate the benefits of our integration with examples, report on experiments and analyze open challenges.

1 Introduction

In recent years much progress has been made regarding applications of first-order automated theorem provers (FO-ATP) in ontology reasoning and question answering. A prominent example is the application of FO-ATPs to the Suggested Upper-Level Ontology (SUMO) [22]. Related work has also been reported for Cyc [24] and the LogAnswer question answering system [13], and further related references are given in [13]. In all those approaches, translations from the ontology representation languages into proper first-order representations are employed.

Challenges that have been identified in this application context, amongst others, include the large theories challenge and answer extraction problem. The former addresses the problem that the proofs are often shallow while the axiom sets are usually huge and may contain lots of information irrelevant to a given query. The latter deals with the issue of extracting single or multiple answers to queries from prover output.

Another challenge, which is in the focus of this paper, is that knowledge bases such as SUMO contain a small but significant amount of higher-order representations. SUMO, for example, began as just an upper level ontology encoded in first-order logic but subsequently its logic has been expanded to also include higher-order elements.

The approach taken in the above systems to reason with higher-order content is to employ specific translation 'tricks', possibly in combination with or in addition to some pre-processing techniques. An example is the quoting technique for embedded formulas as employed in the Sigma knowledge engineering environment [22]. Unfortunately, however, this solution is strongly limited. The effect is that many desirable inferences are currently not supported, so that many queries cannot be answered. Illustrating examples are presented in this paper and a solution is proposed that employs higher-order automated theorem proving (HO-ATP) for the task. Our solution exploits the new TPTP infrastructure for higher-order automated theorem proving [27] and provides a generic translation from the Standard Upper Ontology Knowledge Interchange Format (SUO-KIF) [20] into the new typed higher-order form (THF) language of the TPTP.

This paper is structured as follows. In Section 2 we briefly sketch the background of our work. Section 3 further motivates it with examples. Our translation from the SUO-KIF to TPTP THF is presented in Section 4. Implementation, system integration and experiments are reported in Section 5. The paper ends with a discussion of open challenges and future work.

*This work was funded by the German Research Foundation (DFG) under grant BE 2501/6-1.

2 Background

SUMO [17] is an open source[1], formal ontology. In addition to the expressive logic it was authored in, it has also been translated into the OWL semantic web language. It has undergone ten years of development, review by a community of hundreds of people, and application in expert reasoning, linguistics and performance testing for theorem provers. SUMO has been subjected to partial formal verification with automated theorem provers. This consisted of asking a theorem prover to prove the negation of each axiom in the knowledge base. While necessarily incomplete, this did focus the attention of the prover with more success than simply asking it to prove "false". With repeated testing on incrementally more generous time allotments, this method caught a number of non-obvious contradictions. It has been one method of many partial methods to ensure quality and consistency.

SUMO covers areas of knowledge such as temporal and spatial representation, units and measures, processes, events, actions, and obligations. SUMO has been extended with a number of domain specific ontologies, which are also public, together they number some 20,000 terms and 70,000 axioms. Domain specific ontologies extend and reuse SUMO, for example, in the areas of finance and investment, country almanac information, terrain modeling, distributed computing, and biological viruses. SUMO has also been mapped by hand [18] to the entire WordNet lexicon of approximately 100,000 noun, verb, adjective and adverb word senses, which not only acts as a check on coverage and completeness, but also provides a basis for application to natural language understanding tasks.

SUMO has natural language generation templates and a multi-lingual lexicon that allows statements in SUMO to be automatically paraphrased in multiple natural languages.

The formal language of SUMO is SUO-KIF, a simplified version of the original KIF [14], with extensions for higher-order logic. Since SUO-KIF syntax is rather self-explaining we avoid a formal introduction here and provide some explanations on the fly. For further details we refer to [20].

Sigma [21] is a browsing and inference system that is both a stand-alone system for ontology development and an embeddable component for reasoning. We have also developed a set of optimizations that improve the performance of reasoning on SUMO, typically by "trading space for time" — pre-computing certain inferences and storing them in the knowledge base [21]. In many cases this results in speedups of several orders of magnitude. While Sigma originally included only the Vampire prover for performing logical inference on SUMO, it now embeds the TPTPWorld environment [29], giving it access to some 40 different systems, including the world's most powerful automated theorem provers and model generators. Sigma now also integrates the SInE reasoner [16], which was the winner of the SUMO division of the CASC international theorem proving competition [23]. Use of the SInE axiom selection system has been shown to provide orders of magnitude improvements in theorem proving performance compared to using top-performing theorem prover, such as E or Vampire, alone. By selecting its best guess at axioms relevant to a particular query, it can dramatically reduce the search space for solving queries on large knowledge bases, such as SUMO, where only a small number of axioms are likely to be relevant to any given query. Sigma handles making statements and posing queries to the different reasoners, optimizing the knowledge sent to them to support efficient inference, and handling their output, formatting answers and proofs in a standard and attractive format. Sigma includes a Java API and XML messaging interface.

HO-ATP is currently experiencing a renaissance that has been fostered by the recent extension of the successful TPTP infrastructure for first-order logic [26] to higher-order logic, called

[1] `www.ontologyportal.org`

TPTP THF [27, 28]. Available HO-ATPs include LEO-II [10], TPS [2], IsabelleP, IsabelleM/N[2] and Satallax [3]. These systems are available online via the SystemOnTPTP tool [25], they support the TPTP THF infrastructure, and they employ THF0 [11], the simple type theory fragment of the THF language, as input language.

3 Examples and Challenges

Our goal has been to enable and study applications of HO-ATP for question answering in ontology reasoning, exemplary in SUMO. In this section we present some motivating examples. They illustrate the potential of our approach to reason within temporal and other contexts. We also point to a problem regarding Boolean extensionality and epistemic modalities.

Embedded Formulas Embedded formulas are one prominent source of higher-order aspects in SUMO. This is illustrated by the following example, which has been adapted from [22]. (Premises are marked with **P** and the query is marked with **Q**. In SUMO variables always start with a '`?`'. Free variables in queries are implicitly existentially quantified and those in premises are implicitly universally quantified.)

Example 1 (During 2009 Mary liked Bill and Sue liked Bill. Who liked Bill in 2009?)**.**
P1: `(holdsDuring (YearFn 2009) (and (likes Mary Bill) (likes Sue Bill)))`
Q: `(holdsDuring (YearFn 2009) (likes ?X Bill))`

The challenge is to reason about the embedded formulas `(and (likes Mary Bill) (likes Sue Bill))` and `(likes ?X Bill)` within the context `(holdsDuring (YearFn 2009) ...)`. In our example, the embedded formula in the query does not match the embedded formula in the premise, however, it is inferable from it. The quoting technique presented in [22], which encodes embedded subformulas as strings, fails for this query. There are possible further 'tricks' though which could eventually be applied. For example, we could split **P1** in a pre-processing step into **P2:** `(holdsDuring (YearFn 2009) (likes Mary Bill))` and **P3:** `(holdsDuring (YearFn 2009) (likes Sue Bill))`. However, such means quickly reach their limits when considering more involved embedded reasoning problems. The following modifications of Example 1 illustrate the challenge.

Example 2 (Example 1 modified; 'and' reformulated)**.**
P4:
```
(holdsDuring (YearFn 2009)
             (not (or (not (likes Mary Bill)) (not (likes Sue Bill)))))
```
Q: `(holdsDuring (YearFn 2009) (likes ?X Bill))`

Example 3 (At all times Mary likes Bill. During 2009 Sue liked whomever Mary liked. Is there a year in which Sue has liked somebody?)**.**
P5: `(holdsDuring ?Y (likes Mary Bill))`
P6: `(holdsDuring (YearFn 2009) (forall (?X) (=> (likes Mary ?X) (likes Sue ?X))))`
Q: `(holdsDuring (YearFn ?Y) (likes Sue ?X))`

In particular, Example 3 illustrates that the reasoning tasks may indeed quickly become non-trivial for approaches based on translations to first-order logic. This example can be further modified as follows. Here we use a propositional variable `?P` in order to encode that what generally holds also holds in all `holdsDuring`-contexts.

[2]IsabelleM and IsabelleN are model finders in the Isabelle proof assistant [19] that have been made available in batch mode, while IsabelleP applies a series of Isabelle proof tactics in batch mode.

Example 4 (What holds that holds at all times. Mary likes Bill. During 2009 Sue liked whomever Mary liked. Is there a year in which Sue has liked somebody?)**.**
P7: `(=> ?P (holdsDuring ?Y ?P))`
P8: `(likes Mary Bill)`
P9: `(holdsDuring (YearFn 2009) (forall (?X) (=> (likes Mary ?X) (likes Sue ?X))))`
Q: `(holdsDuring (YearFn ?Y) (likes Sue ?X))`

We may instead of **P7** express that true things hold at all times in an alternative way, cf. **P7'** below.[3]

Example 5 (Example 4 modified)**.**
P7': `(holdsDuring ?Y True)`
P8: `(likes Mary Bill)`
P9: `(holdsDuring (YearFn 2009) (forall (?X) (=> (likes Mary ?X) (likes Sue ?X))))`
Q: `(holdsDuring (YearFn ?Y) (likes Sue ?X))`

Some key steps of the informal argument for the latter query are: Since `True` is always valid and since we assume `(likes Mary Bill)` we know that these two formulas are equivalent. Hence, they are equal. We can thus replace `True` in `(holdsDuring ?Y True)` by `(likes Mary Bill)`. The remaining argument is straightforward.

Set abstraction Another important higher-order construct in SUMO is the set (or class) constructor `KappaFn`. It takes two arguments, a variable and a formula, and returns the set (or class) of things that satisfy the formula. We illustrate the use of `KappaFn` in Example 6.

Example 6 (The number of people John is grandparent of is less than or equal to three. How many grandchildren does John at most have?)**.**
P10: `(<=> (grandchild ?X ?Y) (exists (?Z) (and (parent ?Z ?X) (parent ?Y ?Z))))`
P11: `(<=> (grandparent ?X ?Y) (exists (?Z) (and (parent ?X ?Z) (parent ?Z ?Y))))`
P12: `(lessThanOrEqualTo (CardinalityFn (KappaFn ?X (grandparent John ?X))) 3)`
Q: `(lessThanOrEqualTo (CardinalityFn (KappaFn ?X (grandchild ?X John))) ?Y)`

The query can be proved valid independent of the specific axiomatization of `CardinalityFn`. This is because the embedded set abstractions can be shown equal.

Extensionality In the examples discussed so far we have assumed that the semantics of our logic is classical and that the Boolean and functional extensionality principles are valid. In particular Boolean extensionality, which says that two formulas `P` and `Q` are equal if and only if they are equivalent (or, alternatively, that there are not more than two truth values), is relevant for all of the examples above. Without it we could not even prove the following query since the denotations of the two embedded formulas could be different despite the equivalence of these formulas.

Example 7 (During 2009 Mary liked Bill and Sue liked Bill. Is it the case that in 2009 Sue liked Bill and Mary liked Bill?)**.**
P1: `(holdsDuring (YearFn 2009) (and (likes Mary Bill) (likes Sue Bill)))`
Q: `(holdsDuring (YearFn 2009) (and (likes Sue Bill) (likes Mary Bill)))`

[3] Instead of **P7'** we may equally well use e.g. **P7":** `(holdsDuring ?Y (equal Chris Chris))` or any other embedded tautology.

Functional extensionality, which is required in Example 6 in combination with Boolean extensionality, has been discussed as an option for the semantics of KIF in [15]. The validity of Boolean extensionality has never been questioned though in the literature. Weakening it would require a semantics with more than two truth values and this is not considered an option, neither in [15] nor in [20]. For a detailed discussion of functional and Boolean extensionality in classical higher-order logic we refer to [7].

Modalities Challenge Assuming Boolean extensionality in the semantics of SUO-KIF seems perfectly fine for the above examples. We do not want to conceal, though, the following problem related to it. SUMO employs epistemic modalities, such as `believes` and `knows`. When used in combination with Boolean extensionality, however, inferences are enabled that do obviously contradict their intended meaning. We give an example that is very similar to Example 5; the main difference is that the temporal context has been replaced by an epistemic context.

Example 8 (Adapted Example 5 within epistemic context: Everybody knows that Chris is equal to Chris. Mary likes Bill. Chris knows that Sue likes whomever Mary likes. Does Chris know that Sue likes Bill?)**.**
P7": `(knows ?Y (equal Chris Chris))`
P8: `(likes Mary Bill)`
P9': `(knows Chris (forall (?X) (=> (likes Mary ?X) (likes Sue ?X)))`
Q: `(knows Chris (likes Sue Bill))`

Assuming Boolean extensionality the query is valid, even though we have not explicitly stated the fact `(knows Chris (likes Mary Bill))`. Intuitively, however, this assumption seems mandatory for enabling the proof of the query. Hence, we here (re-)discover an issue that some logicians possibly claim as widely known: modalities have to be treated with great care in classical, extensional higher-order logic. Our ongoing work therefore studies how we can suitably adapt the modeling of affected modalities in SUMO in order to appropriately address this issue.

Relation and Function Variables Generating suitable instantiations for relation or function variables is another prominent higher-order challenge. For instance, in the following query the relation `sib`, with `(sib ?X ?Y)` if and only if `(or (sister ?X ?Y) (brother ?X ?Y)))`, is a valid instantiation for the queried variable `?R`. (There are other instantiations possible for ?R in our example though and enumerating them is a challenge for future work.) Our example illustrates that the invention of new concepts like the notion of sibling from simpler notions like `brother` and `sister` is in principle feasible in higher-order logic, though there are clearly practical limitations.

Example 9 (Mary, Sue, Bill and Bob are mutually distinct. Mary is neither a sister of Sue nor of Bill, and Bob is not a brother of Mary. Sue is a sister of Bill and of Bob, and Bob is a brother of Bill. Is there a relation that holds both between Bob and Bill and between Sue and Bob; we exclude the trivial universal relation $\lambda X, Y.\top$)**.**
P13: `(and (not (equal Mary Sue)) (not (equal Mary Bill)) (not (equal Mary Bob)) (not (equal Sue Bill)) (not (equal Sue Bob)) (not (equal Bob Bill)))`
P14: `(and (not (sister Mary Sue)) (not (sister Mary Bill)) (not (brother Bob Mary)))`
P15: `(and (sister Sue Bill) (sister Sue Bob) (brother Bob Bill))`
Q: `(and (?R Bob Bill) (?R Sue Bob) (not (forall (?X ?Y) (?R ?X ?Y))))`

4 THF Translation

The main objective for our translation from SUMO to TPTP THF0 [11] has been to enable inferences as required for query examples as presented above.

THF0 provides a syntax for Church's simple type theory [1], that is, a classical logic built on top of the simply typed λ-calculus. The standard base types in simple type theory are o and ι; the former denotes the set of Booleans and the latter a (non-empty) set of individuals . They are represented in THF0 as `$i` and `$o`. Further base types can be declared as needed. Function types in THF0 are encoded with the `>`-constructor, e.g. the type of predicates (resp. sets) over type `$i` is denoted as `$i > $o`. THF0 files obey the convention that the types of constant symbols and variable symbols have to be declared before their first use. Type declarations for constant symbols are typically provided in a type signature part at the beginning of each THF0 file while types of variable symbols are provided in their binding positions.

In our translation of SUMO to THF0 we recursively analyze all SUMO terms and subterms with the aim of assigning consistent type information to them. From this process we then extract the assigned type information for all constant and variable symbols as required in THF0 files. When applying our transformation procedure to **P12**, for example, we generate the following THF0 information:

```
%%% The extracted Signature %%%
thf(grandparent_THFTYPE_IiioI,type,(
    grandparent_THFTYPE_IiioI: $i > $i > $o )).

thf(lCardinalityFn_THFTYPE_IIioIiI,type,(
    lCardinalityFn_THFTYPE_IIioIiI: ( $i > $o ) > $i )).

thf(lJohn_THFTYPE_i,type,(
    lJohn_THFTYPE_i: $i )).

thf(ltet_THFTYPE_IiioI,type,(
    ltet_THFTYPE_IiioI: $i > $i > $o )).

thf(n3_THFTYPE_i,type,(
    n3_THFTYPE_i: $i )).

%%% The translated axioms %%%
thf(ax,axiom,
    ( ltet_THFTYPE_IiioI
    @ ( lCardinalityFn_THFTYPE_IIioIiI
      @ ^ [X: $i] :
          ( grandparent_THFTYPE_IiioI @ lJohn_THFTYPE_i @ X ) )
    @ n3_THFTYPE_i )).
```

This THF0 representation is, for obvious reasons, not intended for human consumption. It serves the sole purpose of communicating the reasoning problem to the higher-order theorem provers. We briefly sketch a few aspects: (i) So far, we use THF0 type `$i` as only base type other than `$o`; for example, SUMO formulas and sentences are mapped to type `$o` while constants such as `lJohn_THFTYPE_i` and `n3_THFTYPE_i`, which are the translations of the

SUMO constants `John` and `3`, are currently both declared of type `$i`.[4] Function types, e.g. for `lCardinalityFn_THFTYPE_IIioIiI`, are determined by our translation algorithm. Future work includes the introduction of further base types in combination with a better exploitation of the rich typing information already available in SUMO. (ii) As expected, the simple type computed for `lCardinalityFn_THFTYPE_IIioIiI`[5], the translation of SUMO constant `CardinalityFn`, is `( $i > $o ) > $i`, that is, the arguments for this constant have to be sets of objects of type `$i`. (iii) `KappaFn` is mapped to λ-abstraction.

Assigning types to SUMO terms is in fact not as straightforward as this example might suggest. One major problem is that SUMO supports self-applications as in the following SUMO axiom.

```
(instance instance BinaryRelation)
```

In order to translate such axioms we currently split affected constants like `instance` into separate constants:

```
%%% The extracted Signature %%%
thf(lBinaryPredicate_THFTYPE_i,type,(
    lBinaryPredicate_THFTYPE_i: $i )).

thf(instance_THFTYPE_IIiioIioI,type,(
    instance_THFTYPE_IIiioIioI: ( $i > $i > $o ) > $i > $o )).

thf(instance_THFTYPE_IiioI,type,(
    instance_THFTYPE_IiioI: $i > $i > $o )).

%%% The translated axiom(s) %%%
thf(ax,axiom,
    ( instance_THFTYPE_IIiioIioI @ instance_THFTYPE_IiioI
                                 @ lBinaryPredicate_THFTYPE_i )).
```

Obviously, we thereby lose important information, for example, in our examples we now only know that `instance_THFTYPE_IiioI` denotes a binary relation. If we want this information restored for `instance_THFTYPE_IIiioIioI` we can generate a new constant `instance_THFTYPE_IIIiioIioIioI` and a new axiom

```
thf(ax,axiom,((instance_THFTYPE_IIIiioIioIioI @ instance_THFTYPE_IIiioIioI
                                              @ lBinaryPredicate_THFTYPE_i))).
```

Currently such a duplication of axioms is still disabled in our translation. Future work, however, will study the need for such duplications more closely.

Our first project goal has thus been achieved, namely to provide a translation of the entire SUMO into THF0 that can be parsed and type checked by all THF0 reasoners in the TPTP and that, in spite of its need for further improvement, can already serve as a starting point for examples as we have discussed.[6]

[4] TPTP syntax requires all constants in lower case, hence, the leading 'l' and 'n'. Moreover, we also encode the computed type information in the constant name; the reasons for this will become clear below.

[5] The 'I's encode bracketing information.

[6] The THF0 translation of SUMO is available at: `http://www.ags.uni-sb.de/~chris/papers/SUMO.thf`.

5 System Implementation, Integration, and Initial Experiments

The THF translation mechanism has been implemented as part of the Sigma environment. This enabled the reuse of already existing infrastructure, e.g. for manipulating formulas and knowledge bases, as well as the reuse of existing first-order logic TPTP tools in Sigma.

Additionally, an initial integration of the LEO-II system has been created with Sigma. There are three modes in which LEO-II can be applied to queries in this integration. The *local* mode only translates the user assertions and the query, the *global* mode translates the entire SUMO knowledge base and then adds the user assertions and the query, and the *SInE* mode employs Hoder's SInE system to extract a (hopefully) relevant subset of the axioms from the SUMO knowledge base.

We have conducted initial experiments with the LEO-II prover (version v1.1) integrated to Sigma: All examples in this paper can be effectively solved by LEO-II in local mode, except for Example 9: Ex.1 (0.19s), Ex.2 (0.19s), Ex.3 (0.13s), Ex.4 (0.16s), Ex.5 (0.08s), Ex.6 (0.34), Ex.7 (0.18s), Ex.8 (0.04s), Ex.9 (2642.55s) — the timings were obtained on a standard MacBook Pro with a 2.4 GHz Intel Core 2 Duo processor and 2GB of memory. There is actually no general problem with Example 9, only LEO-II performs particularly poorly on it and the reasons for this should be investigated. Tests with other HO-ATPs via the SystemOnTPTP tool confirm that IsabelleP, for example, finds a proof in 10s.

We have submitted 28 related examples, each in two or three different versions, to the TPTP for inclusion. The different versions are corresponding to the three modes for calling LEO-II in Sigma as discussed before. Recent experiments of Geoff Sutcliffe with his TPTP infrastructure indicates that LEO-II is slightly ahead of the other provers for these example problems. The important news, however, is that the main hypotheses of our work has been confirmed: higher-order automated reasoners have the potential to advance the state-of-art in ontology reasoning and question answering. This has also been confirmed by the detection (and subsequent fixing) of some problematic axioms in SUMO in the course of our experiments. For example, in the following axiom for 'pretending' the last occurrence of `True` has been detected as semantically wrong and was subsequently replaced by `False` ('pretending' is is a social interaction where a cognitive agent or group of cognitive agents attempts to make another cognitive agent or group of cognitive agents believe something that is false):

```
(=> (instance ?PRETEND Pretending)
    (exists (?PERSON ?PROP) (and (hasPurpose ?PRETEND (believes ?PERSON ?PROP))
                                 (truth ?PROP True))))
```

Moreover, not only HO-ATP theorem provers are applicable to support ontology reasoning but also higher-order model finders. The IsabelleN model finder, for example, has revealed several typos in earlier versions of our example problems by constructing countermodels.

On the downside, however, our tests also show that much further work is needed for turning our proof of concept into a practically reliable and robust success story. For example, only very few of our examples can currently be solved in the SInE mode and even less can be solved in the global mode. Hence, the challenges involved in making inference efficient over large theories turns out even worse for the HO-ATPs than it already is for the FO-ATPs. This was to be expected though, in particular, since the theoretical and technical maturity of HO-ATPs is still many years, if not decades, behind those of FO-ATP systems.

6 Discussion

In this paper we have shown that HO-ATP is in principle capable of advancing the state-of-art in ontology reasoning and question answering in expressive frameworks such as SUMO. There are many open issues though that require much further thought and work. We briefly discuss a few.

The large theories challenge requires the development or adaptation of strong relevance filters such as SInE. We are still using an older version of SInE and we speculate that the latest version, in which the maximal number of selected axioms can be predetermined by a parameter, may already significantly improve the performance of the HO-ATPs in SInE mode.

There is also an important meta-reasoning task to be solved for Sigma. Currently, the selection of reasoners and further options is task of the user. In the future, however, we plan to automate this task. We envision distributed, possibly even cooperative, proof attempts by reasoners working for different translation targets like TPTP FOL and TPTP THF. Hence, the intended meta-reasoner needs to support various non-trivial tasks including: (i) selection of appropriate reasoners and translation targets, (ii) relevance filtering, (iii) control of (distributed) prover execution, (iv) extraction of answers from prover results, (v) result verification, (vi) preparation of answers and their presentation to the user.

For several of these tasks existing technology can possibly be adapted. Answer extraction, for example, is already supported in Sigma for all first-order provers which obey the standardized TPTP proof output format. And for supporting distributed or even cooperative reasoning with external systems in Sigma the agent-based OANTS architecture [12] can possibly be adapted.

One of the most interesting and relevant challenges, however, is the modalities challenge. As we have shown, Boolean extensionality and epistemic modalities such 'knows' or 'believes', for example, do not go well together. This observation is relevant beyond the borders of SUMO and it clearly also affects the current first-order translations: if they will eventually be extended so that they can successfully handle Examples 4 and 5, then they will also face the problem of Example 8.

Traditional (propositional) modal logics approaches and reasoners seem hardly applicable for the task since the modalities are usually employed in SUMO in combination with other first-order and higher-order constructs. One of our current research directions therefore aims at exploiting recent results that show how (multi-)modal logics can be elegantly encoded as simple fragments of higher-order logics [9, 8]. The idea is to consider modalities such as 'knows' or 'believes' as abbreviations for lambda-terms (as presented in [9]) denoting the appropriate modal operators. This solution explicitly supports the coexistence of different modalities in combination with other first-order and higher-order constructs. Related case studies on epistemic reasoning (for example, an elegant and efficient solution of the Wise Men Puzzle) with classical, extensional higher-order theorem provers can be found in [6].

We may also consider a modification of the theorem prover LEO-II and its underlying calculus. The idea would be to provide means for annotating function and predicate symbols regarding their pre-determined extensionality properties and to distinguish in the inference process according to these annotations. `holdsDuring`, for example, would be annotated as fully extensional, while `knows` and `believes` would not. Hence, the inference in Example 8 could be blocked in the prover while Example 5 would still go through. A respective research proposal in such a direction can already be found in [5] (which unfortunately was not funded at the time). Such a solution would allow us to make an informed and context-dependent choice regarding the extensionality principles for the semantics of SUO-KIF.

References

[1] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer Academic Publishers, second edition, 2002.

[2] Peter B. Andrews and Chad E. Brown. TPS: A hybrid automatic-interactive system for developing proofs. *Journal of Applied Logic*, 4(4):367–395, 2006.

[3] Julian Backes and Chad E. Brown. Analytic tableaux for higher-order logic with choice. In J. Giesl and R. Haehnle, editors, *IJCAR 2010 - 5th International Joint Conference on Automated Reasoning*, LNAI, Edinburgh, UK, July 2010. Springer. to appear.

[4] P. Baumgartner, A. Armando, and D. Gilles, editors. *Proceedings of the 4th International Joint Conference on Automated Reasoning*, number 5195 in Lecture Notes in Artificial Intelligence. Springer, 2008.

[5] Christoph Benzmüller. ALONZO: Deduktionsagenten höherer Ordnung für Mathematische Assistenzsysteme. Research project proposal to the DFG Aktionsplan Informatik, available at `http://www.ags.uni-sb.de/~chris/papers/R23.pdf`, 2003.

[6] Christoph Benzmüller. *Automating Quantified Multimodal Logics in Simple Type Theory – A Case Study*. SEKI Working-Paper SWP-2009-02 (ISSN 1860-5931)). SEKI Publications, DFKI Bremen GmbH, Germany, 2009. `http://arxiv.org/abs/0905.4369`.

[7] Christoph Benzmüller, Chad Brown, and Michael Kohlhase. Higher-order semantics and extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004.

[8] Christoph Benzmüller and Lawrence C. Paulson. *Quantified Multimodal Logics in Simple Type Theory*. SEKI Report SR–2009–02 (ISSN 1437-4447). SEKI Publications, DFKI Bremen GmbH, Germany, 2009. `http://arxiv.org/abs/0905.2435`.

[9] Christoph Benzmüller and Lawrence C. Paulson. Multimodal and intuitionistic logics in simple type theory. *The Logic Journal of the IGPL*, 2010. In print.

[10] Christoph Benzmüller, Lawrence C. Paulson, Frank Theiss, and Arnaud Fietzke. LEO-II — a cooperative automatic theorem prover for higher-order logic. In Baumgartner et al. [4], pages 162–170.

[11] Christoph Benzmüller, Florian Rabe, and Geoff Sutcliffe. THF0 — the core TPTP language for classical higher-order logic. In Baumgartner et al. [4], pages 491–506.

[12] Christoph Benzmüller and Volker Sorge. OANTS – an open approach at combining interactive and automated theorem proving. In M. Kerber and M. Kohlhase, editors, *Symbolic Computation and Automated Reasoning*, pages 81–97. A.K.Peters, 2000.

[13] Ulrich Furbach, Ingo Glöckner, and Björn Pelzer. An application of automated reasoning in natural language question answering. *AI Communications*, 23(2-3):241–265, 2010. PAAR Special Issue.

[14] Michael R. Genesereth. Knowledge interchange format. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, pages 238–249. Morgan Kaufmann, 1991.

[15] Patrick Hayes and Christopher Menzel. A semantics for the knowledge interchange format. In *IJCAI 2001 Workshop on the IEEE Standard Upper Ontology*, August 2001.

[16] Krystof Hoder. Automated reasoning in large knowledge bases. Master's thesis, Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague, 2008.

[17] Ian Niles and Adam Pease. Towards a standard upper ontology. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9, New York, NY, USA, 2001. ACM.

[18] Ian Niles and Adam Pease. Linking lexicons and ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In H. R. Arabnia, editor, *Proceedings of the International Conference on Information and Knowledge Engineering. IKE'03, June 23 - 26, 2003, Las Vegas, Nevada, USA, Volume 2*, pages 412–416. CSREA Press, 2003.

[19] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for*

Higher-Order Logic, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[20] Adam Pease. Standard Upper Ontology Knowledge Interchange Format. `http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/sigma/suo-kif.pdf`.

[21] Adam Pease. The Sigma ontology development environment. In F. Giunchiglia, A. Gomez-Perez, A. Pease, H. Stuckenschmid, Y. Sure, and S. Willmott, editors, *Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, volume 71. CEUR Workshop Proceedings, 2003.

[22] Adam Pease and Geoff Sutcliffe. First order reasoning on a large ontology. In G. Sutcliffe, J. Urban, and S. Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[23] Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. Large theory reasoning with SUMO at CASC. *AI Communications*, 23(2-3):137–144, 2010.

[24] Deepak Ramachandran, Pace Reagan, and Keith Goolsbey. First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology. In Shvaiko P., editor, *Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications, Pittsburgh, Pennsylvania, USA, 2005*. Technical Report WS-05-01 published by The AAAI Press, Menlo Park, California, July 2005.

[25] Geoff Sutcliffe. TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, *Proceedings of the 2nd International Computer Science Symposium in Russia*, number 4649 in Lecture Notes in Computer Science, pages 7–23. Springer, 2007.

[26] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[27] Geoff Sutcliffe and Christoph Benzmüller. Automated reasoning in higher-order logic using the TPTP THF infrastructure. *Journal of Formalized Reasoning*, 3(1):1–27, 2010.

[28] Geoff Sutcliffe, Christoph Benzmüller, Chad Brown, and Frank Theiss. Progress in the development of automated theorem proving for higher-order logic. In R. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2009.

[29] Steven Trac, Geoff Sutcliffe, and Adam Pease. Integration of the TPTPWorld into SigmaKEE. In B. Konev, R. Schmidt, and S. Schulz, editors, *PAAR/ESHOL*, volume 373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.