Higher-Order Aspects and Context in SUMO

Christoph Benzmüller[1,*]

Freie Universität Berlin, Germany

Adam Pease

Articulate Software, Angwin, CA, USA

Abstract

This article addresses the automation of higher-order aspects in expressive ontologies such as the Suggested Upper Merged Ontology SUMO. Evidence is provided that modern higher-order automated theorem provers like LEO-II can be fruitfully employed for the task. A particular focus is on embedded formulas (formulas as terms), which are used in SUMO, for example, for modeling temporal, epistemic, or doxastic contexts. This modeling is partly in conflict with SUMO's assumption of a bivalent, classical semantics and it may hence lead to counterintuitive reasoning results with automated theorem provers in practice. A solution is proposed that maps SUMO to quantified multimodal logic which is in turn modeled as a fragment of classical higher-order logic. This way automated higher-order theorem provers can be safely applied for reasoning about modal contexts in SUMO.

Our findings are of wider relevance as they analogously apply to other expressive ontologies and knowledge representation formalisms.

Keywords: expressive ontologies, context, classical higher-order logic, Boolean extensionality, quantified multimodal logic, automated theorem proving, relevance filtering

1. Introduction

Expressive ontologies such as the Suggested Upper Merged Ontology SUMO [1, 2] or Cyc [3] contain a small but significant number of higher-order representations.

This article investigates higher-order aspects in the SUMO ontology with the aim to improve the automation support for such aspects in practice. The particular focus is on embedded formulas (formulas as terms), which are employed in SUMO, for example, for modeling temporal, epistemic, or doxastic contexts.

The basic idea for modeling contexts in SUMO is simple. A statement like (*loves Bill Mary*) is restricted, for instance, to the year 2009 by wrapping it (at subterm level) into respective context information:

(*holdsDuring*
  (*YearFn 2009*)
  (*loves Bill Mary*))

Similarly, the statement can be put into an epistemic or doxastic context:

(*knows/believes*
  Ben
  (*loves Bill Mary*))

Moreover, contexts can be flexibly combined:

(*believes*
  Bill
  (*knows*
    Ben
    (*loves Bill Mary*)))

Contexts have been discussed in the literature as a means to achieve both generality [4, 5] and locality [6]. Flexible nestings of contexts, as illustrated above, support the generality aspect. The locality aspect, which calls for a separation of the knowledge that is relevant in a given situation from all available knowledge, is also addressed in our work. The technique adopted for this is relevance filtering, that is, the goal directed selection of axioms from a large knowledge base.

The work presented in this article is pioneering the application of higher-order automated theorem proving to expressive ontologies like SUMO. Since this entails the automation of embedded formulas it also entails reasoning with contexts. Moreover, as part of our work we reveal and subsequently fix a problem in SUMO that has been unnoticed before: some modal contexts are in conflict with the assumption of a bivalent[2], clas-

*Corresponding author

Email addresses: c.benzmueller@googlemail.com (Christoph Benzmüller), apease@articulatesoftware.com (Adam Pease)

URL: http://christoph-benzmueller.de/ (Christoph Benzmüller), http://www.adampease.org/professional/ (Adam Pease)

[1]Funded by the German Research Foundation under grants BE 2501/6-1 and BE 2501/8-1.

[2]Bivalence expresses that there are exactly two truth values. This aspect of

sical semantics.

Our findings are not exclusive for SUMO and they analogously apply to other expressive knowledge representation frameworks, in particular to McCarthy's pioneering work [5] and its descendants.

The structure of the article is as follows. Section 2 introduces SUMO and presents some background information on higher-order logic and on higher-order theorem proving. Section 3 provides an overview on higher-order aspects in SUMO using small examples. Moreover, the mentioned conflict between SUMO's modeling of modal contexts and SUMO's implicit assumption of a bivalent, classical semantics is discussed. In Section 4 a mapping from SUMO's SUO-KIF representation language [7] to the TPTP THF0 language [8, 9] is presented. TPTP THF0 is a practical syntax format for classical higher-order logic that enables the application of various off-the-shelf higher-order theorem provers. We have exploited this mapping in some experiments with our running examples. These experiments, which are reported in Section 5, provide first evidence that higher-order automated reasoning in SUMO is useful and feasible in practice. In particular the prover LEO-II [10], which now also supports relevance filtering, appears suited for the task. In Section 6 we present a solution for SUMO's conflict with Boolean extensionality, which we revealed in Section 3. The solution is to translate SUMO into quantified multimodal logic which is in turn modeled and mechanized as a fragment of THF0.

2. Preliminaries

2.1. The Suggested Upper Merged Ontology SUMO

SUMO [2] is an open source[3], 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 first-order 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. The upper ontology contains about 4000 axioms. SUMO has been extended with a mid-level ontology and 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 [11] 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. Moreover, SUMO has recently been extended by large factbases of millions of statements, including YAGO [12].

SUMO has natural language generation templates and a multilingual 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 [13], 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 [7].

Sigma [14] is a browsing and inference system that is both a stand-alone system for ontology development and an embeddable component for reasoning. We have 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 [14]. In many cases this results in speedups of several orders of magnitude. While Sigma originally included only the Vampire prover [15] for performing logical inference on SUMO, it now embeds the TPTPWorld environment [16], giving it access to some 40 different systems, including the world's most powerful automated theorem provers and model generators. Sigma integrates the SInE reasoner [17], which was the winner of the SUMO division of the CASC international theorem proving competition [18]. 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 [19] 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.

2.2. Higher-Order Logic and Higher-Order Theorem Proving

There are many quite different frameworks that fall under the general label of "higher-order logic". The notion reaches back to Frege's original predicate calculus [20]. Inconsistencies in Frege's system, caused by the circularity of constructions such as "the set of all sets that do not contain themselves", made it clear that the expressivity of the language had to be restricted in some way. One line of development, which became the traditional route for mathematical logic, and which is not

classical logics is also addressed by the notion of Boolean extensionality, cf. Section 3.3. In the remainder of this article we use both notions synonomously.

[3] www.ontologyportal.org

addressed further here, is the development of axiomatic first-order set theories, e.g. Zermelo-Fraenkel set theory. Russell suggested using type hierarchies, and worked out ramified type theory. Church (inspired by work of Carnap) later introduced simple type theory [21], a higher-order framework built on his simply typed λ-calculus, employing types to reduce expressivity and to remedy paradoxes and inconsistencies. Simple type theory is often also called classical higher-order logic (HOL).

Via the Curry-Howard isomorphism, typed λ-calculi can also be exploited to encode proofs as types. The simply typed λ-calculus, for example, is sufficient for encoding propositional logic. More expressive logics can be encoded using dependent types and polymorphism [22, 23, 24]. In combination with Martin Löf's intuitionistic theory of types [25], originally developed for formalizing constructive mathematics, this research led the foundations of modern type theory.

During the last decades various proof assistants have been built for both classical higher-order logic and type theory. Prominent interactive provers for classical higher logic include HOL [26], HOL Light [27], PVS [28], Isabelle/HOL [29], and OMEGA [30]. Prominent interactive type theory provers include the pioneering Automath system [31], Nuprl [32], Lego [33], Matita [34], and Coq [35]. The latter three are based on the calculus of constructions [36]. Further type theory systems are the logical frameworks Elf [37] and Twelf [38].

Automation of HOL has been pioneered by the work of Andrews on resolution in type theory [39], by Huet's pre-unification algorithm [40] and his constrained resolution calculus [41], and by Jensen and Pietroswski's [42] work. More recently extensionality and equality reasoning in HOL has been studied [43, 44, 45, 46]. The TPS system [47, 48], which is based on a higher-order mating calculus, is a pioneering ATP system for HOL.

The automation of HOL currently experiences a renaissance that has been fostered by the recent extension of the successful TPTP infrastructure for first-order logic [49] to higher-order logic, called TPTP THF [9, 50]. THF0, which is a concrete syntax for HOL, is the starting point for the development of more expressive languages in the THF family. Meanwhile several higher-order provers and model finders accept the THF0 language as input. These systems are available online via the SystemOnTPTP tool [51], where they can be easily employed for experiments without need for local installations. As a result of our work all of these systems are now applicable to SUMO. We briefly describe these THF0 reasoners in more detail (their descriptions are adapted from [9]):

LEO-II. LEO-II [10], the successor of LEO [52], is an automated theorem prover for HOL which is based on extensional higher-order resolution [43]. More precisely, LEO-II employs a refinement of extensional higher-order RUE resolution [44]. LEO-II is designed to cooperate with specialist systems for fragments of HOL; this was motivated by findings in previous work [53]. By default, LEO-II cooperates with the first-order ATP systems E [54]. LEO-II is often too weak to find a refutation amongst the steadily growing set of clauses on its own. However, some of the clauses in LEO-II's search space attain a special status: they are first-order clauses modulo the application of an appropriate transformation function. The default transformation is Hurd's fully typed translation [55]. Therefore, LEO-II launches a cooperating first-order ATP system every n iterations of its (standard) resolution proof search loop (e.g., n = 10). If the first-order ATP system finds a refutation, it communicates its success to LEO-II, which causes LEO-II to terminate and to report overall success. Communication between LEO-II and the cooperating first-order ATP system uses the TPTP language and standards.

TPS. TPS is a pioneering higher-order theorem proving system [47, 48]. It can be used to prove theorems of HOL automatically, interactively, or semi-automatically. When searching for a proof automatically, TPS first searches for an expansion proof [56] or an extensional expansion proof [46] of the theorem. Part of this process involves searching for acceptable matings [57]. Using higher-order unification, a pair of occurrences of subformulas (which are usually literals) is mated appropriately on each vertical path through an expanded form of the theorem to be proved. The behavior of TPS is controlled by hundreds of flags. A set of flags, with values for them, is called a mode. Forty-nine modes have been found that collectively cover the automation power of TPS. As the modes have quite different capabilities, and it is expected that any proofs found by any mode will be found quickly, strategy scheduling the modes is a simple way of obtaining greater coverage. A Perl script has been used to do this, running selected modes for a specified amount of time.

Satallax. Satallax [58] is a higher-order automated theorem prover with additional model finding capabilities. The system is based on a complete ground tableau calculus for HOL with a choice operator [59]. An initial tableau branch is formed from the axioms of the problem and negation of the conjecture (if any is given). From this point on, Satallax tries to determine unsatisfiability or satisfiability of this branch. Satallax progressively generates higher-order formulas and corresponding propositional clauses. These formulas and propositional clauses correspond to instances of the tableau rules. Satallax uses the SAT solver MiniSat as an engine to test the current set of propositional clauses for unsatisfiability. If the clauses are unsatisfiable, then the original branch is unsatisfiable. If there are no quantifiers at function types, the generation of higher-order formulas and corresponding clauses may terminate. In such a case, if MiniSat reports the final set of clauses as satisfiable, then the original set of higher-order formulas is satisfiable (by a standard model in which all types are interpreted as finite sets).

Isabelle. The higher-order proof assistant Isabelle/HOL [29] is normally used interactively. In this mode it is possible to apply various automated tactics that attempt to solve the current goal without further user interaction. Examples of these tactics are *blast*, *auto*, and *metis*. It is also possible to run Isabelle from the command line, passing in a theory file containing a lemma to prove. Finally, Isabelle theory files can include ML code to

be executed when the file is processed. While it was probably never intended to use Isabelle as a fully automatic system, these three features have been combined to implement a fully automatic Isabelle/HOL. The TPTP2X Isabelle format module outputs a THF problem in Isabelle/HOL syntax, augmented with ML code that runs tactics in sequence, each with a CPU time limit until one succeeds or all fail.

Refute and Nitpick. The ability of Isabelle to find models or countermodels using the *refute* [60] and *nitpick* [61] commands has also been integrated into automatic systems. This provides the capability to find models for THF0 formulas, which confirm the satisfiability of axiom sets, or the countersatisfiability of non-theorems. This has been particularly useful for exposing errors in some THF0 problem encodings, and revealing bugs in the THF0 theorem provers (and conversely, the theorem provers have been useful in debugging Refute and Nitpick).

3. Higher-Order Aspects in SUMO – Examples

Our goal has been to enable and study applications of higher-order automated theorem proving for reasoning in expressive ontologies, exemplary in SUMO. In this section we present and discuss some motivating examples.

3.1. Embedded Formulas and Context

Embedded formulas are one prominent source of higher-order aspects in SUMO. This is illustrated by the following example, which has been adapted from [62]. (Premises are prefixed with *P* and the query is prefixed 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?)**.**

```
(holdsDuring                          (P1.1)
  (YearFn 2009)
  (and
    (likes Mary Bill)
    (likes Sue Bill)))
____
(holdsDuring                          (Q1)
  (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 temporal 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 first-order quoting technique for reasoning with such embedded formulas presented by Pease and Sutcliffe [62], which encodes embedded formulas as strings, fails for this query. There are possible further "tricks" though which could eventually be applied. For example, we could split P1.1 in a pre-processing step into

(*holdsDuring* (*YearFn 2009*) (*likes Mary Bill*))

and

(*holdsDuring* (*YearFn 2009*) (*likes Sue Bill*))

However, such simple 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)**.**

```
(holdsDuring                          (P2.1)
  (YearFn2009)
  (not
    (or
      (not (likes Mary Bill))
      (not (likes Sue Bill)))))
____
(holdsDuring                          (Q2)
  (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?)**.**

```
(holdsDuring                          (P3.1)
  ?Y
  (likes Mary Bill))
(holdsDuring                          (P3.2)
  (YearFn 2009)
  (forall (?X)
    (=>
      (likes Mary ?X)
      (likes Sue ?X))))
____
(holdsDuring                          (Q3)
  (YearFn ?Y)
  (likes Sue ?X))
```

The embedded quantified formula in Example 3 well illustrates that the reasoning tasks may 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 universal propositional variable ?*P* in order to encode that what generally holds also holds in all *holdsDuring*-contexts.

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?)**.**

```
(=>                                          (P4.1)
   ?P
   (holdsDuring ?Y ?P))
(likes Mary Bill)                            (P4.2)
(holdsDuring                                 (P4.3)
   (YearFn 2009)
   (forall (?X)
      (=>
         (likes Mary ?X)
         (likes Sue ?X))))
____
(holdsDuring                                 (Q4)
   (YearFn ?Y)
   (likes Sue ?X))
```

We may instead of P4.1 express that true things hold at all times in an alternative way, cf. P5.1 below.

Example 5 (Example 4 modified)**.**

```
(holdsDuring ?Y True)                        (P5.1)
(likes Mary Bill)                            (P5.2)
(holdsDuring                                 (P5.3)
   (YearFn 2009)
   (forall (?X)
      (=>
         (likes Mary ?X)
         (likes Sue ?X))))
____
(holdsDuring                                 (Q5)
   (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*). Now the query easily follows.

Note that instead of P5.1 we may equally well use (*holdsDuring* ?*Y* (*equal Chris Chris*)) or any other embedded tautology.

3.2. Set Abstraction

Another higher-order construct used 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?)**.**

```
(<=>                                         (P6.1)
   (grandchild ?X ?Y)
   (exists (?Z)
      (and
         (parent ?Z ?X)
         (parent ?Y ?Z))))
(<=>                                         (P6.2)
   (grandparent ?X ?Y)
   (exists (?Z)
      (and
         (parent ?X ?Z)
         (parent ?Z ?Y))))
(lessThanOrEqualTo                           (P6.3)
   (CardinalityFn
      (KappaFn ?X
         (grandparent John ?X)))
   3)
____
(lessThanOrEqualTo                           (Q6)
   (CardinalityFn
      (KappaFn ?X
         (grandchild ?X John)))
   ?Y)
```

This query can easily be proved valid independent of the specific axiomatizations of *CardinalityFn* and *lessThanOrEqualTo*, since the two embedded set abstractions can be shown equal with the help of axioms P6.1 and P6.2.

3.3. Extensionality

In the examples discussed so far we have (silently) assumed that the semantics of our logic is classical and that the Boolean and functional extensionality principles are valid. While functional extensionality has actually been discussed as an option for the semantics of KIF [63], the validity of Boolean extensionality has never been questioned though in the literature for KIF and SUO-KIF.

We briefly illustrate the case of Boolean extensionality. For a detailed discussion of functional and Boolean extensionality in classical higher-order logic we refer to Benzmüller, Brown and Kohlhase [45].

Boolean extensionality expresses that two formulas P and Q are equal if and only if they are equivalent, in SUO-KIF syntax:

```
(<=>
   (<=> ?P ?Q)
   (equal ?P ?Q))
```

The left to right direction says that there are not more than two truth values, respectively that whenever two formulas A and B can be shown equivalent then their denotations must be the same, namely either *true* or *false*. Logics with exactly two truth values are also called bivalent logics. Once we have

established equivalence between formulas A and B in a bivalent logic, then, in any formula C in this logic, we may substitute occurrences of A by B (and vice versa). The important aspect is that this principle not only applies to occurrences of A or B at formula level but also to occurrences at term level. For example, (*and* (*likes Mary Bill*) (*likes Sue Bill*)) and (*and* (*likes Sue Bill*) (*likes Mary Bill*)) are obviously equivalent, and hence, by Boolean extensionality, they have identical denotations. Thus, they can always be substituted by each other, also in the term level positions of the following example.

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?)**.**

```
(holdsDuring                          (P7.1)
   (YearFn 2009)
   (and
      (likes Mary Bill)
      (likes Sue Bill)))
____
(holdsDuring                          (Q7)
   (YearFn 2009)
   (and
      (likes Sue Bill)
      (likes Mary Bill)))
```

If Boolean extensionality is not postulated then this substitution principle is blocked. The reason is that we may well consider more than two truth values, for example, $true^1$ and $true^2$ and $false^1$ and $false^2$. In such a situation we could, for example, map the denotation of (*and* (*likes Mary Bill*) (*likes Sue Bill*)) to $true^1$ and the denotation of (*and* (*likes Sue Bill*) (*likes Mary Bill*)) to $true^2$. We could still consider both formulas as equivalent, since both denote a representative of truth. But obviously the formulas no longer have identical denotations. Hence, they can no longer be substituted one by another in term level positions.

The examples so far have been chosen to raise the impression that Boolean extensionality is a natural and useful requirement for SUO-KIF and SUMO. However, this is not the case in general as we will discuss next.

3.4. Boolean Extensionality is in Conflict with Modal Contexts

Boolean extensionality seems fine for the temporal contexts of our previous examples. However, it leads to counterintuitive inferences when applied in other contexts. We illustrate this for epistemic and doxastic contexts. When Boolean extensionality is assumed for either of these contexts, inferences are enabled that do obviously contradict our intuition. 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?)**.**

```
(knows                                (P8.1)
   ?Y
   (equal Chris Chris))
(likes Mary Bill)                     (P8.2)
(knows                                (P8.3)
   Chris
   (forall (?X)
      (=>
         (likes Mary ?X)
         (likes Sue ?X))))
____
(knows                                (Q8)
   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, assuming that Chris actually knows that Mary likes Bill 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, bivalent logics. In general, there is a relation to the question whether we are willing to accept the following principles as theorems (where <*tautology*> stands for an arbitrary tautology):

```
(=>                                   (A)
   (and
      ?PROP
      (holdsDuring ?TIME <tautology>))
   (holdsDuring ?TIME ?PROP))
(=>                                   (B,C)
   (and
      ?PROP
      (knows/believes ?AGENT <tautology>))
   (knows/believes ?AGENT ?PROP))
```

While principle A appears acceptable[4] (even in the stronger form (=> ?*PROP* (*holdsDuring* ?*TIME* ?*PROP*)), the principles B and C are clearly counterintuitive.

In Section 6 we therefore adapt the modeling of affected modalities in SUMO in order to appropriately address this conflict with Boolean extensionality.

3.5. Relation and Function Variables

Relation and function variables are another prominent higher-order challenge in SUMO. For example, the following query asks about a relation ?*R* that holds between *Bob* and *Bill* and between *Sue* and *Bob*. A possible answer in the given situation is the sibling relation, that is, the relation

[4] However, if our interest is in an appropriate modeling of temporal granularity, then we might even want to reject principle A.

(*or* (*sister* ?*X* ?*Y*) (*brother* ?*X* ?*Y*)).[5] The automated synthesis of complex relations (and functions) from more basic, already given ones is generally possible in higher-order order automated theorem provers, though there are still many 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)**.**

(*and* (P9.1)
(*not* (*equal Mary Sue*))
(*not* (*equal Mary Bill*))
(*not* (*equal Mary Bob*)))
(*not* (*equal Sue Bill*))
(*not* (*equal Sue Bob*))
(*not* (*equal Bob Bill*)))

(*and* (P9.2)
(*not* (*sister Mary Sue*))
(*not* (*sister Mary Bill*))
(*not* (*brother Bob Mary*)))

(*and* (P9.3)
(*sister Sue Bill*)
(*sister Sue Bob*)
(*brother Bob Bill*))

——

(*and* (Q9)
(?*R Bob Bill*)
(?*R Sue Bob*))

A first-order approach for reasoning with predicate and function variables has been proposed and implemented for SUMO [62, 64]. This approach is based on some practically motivated restrictions in the search for instantiations of predicate and function variables. More concretely, the search for possible instantiations is restricted to already known concepts in SUMO. The synthesis of the sibling relation above is an example which is already beyond the capabilities of this first-order approach.

4. Mapping SUMO to Classical Higher-Order Logic

A main objective of our work has been to support automation of queries in SUMO as discussed above. In order to enable the application of off-the-shelf higher-order automated theorem provers and model finders for the task we have realized a translation from SUMO's SUO-KIF language into the TPTP THF0 language, which is a syntax for HOL.

4.1. THF0 — A Syntax for HOL

The language HOL is defined by (where $\alpha, \beta, o \in \mathcal{T}$; the set of simple types $\mathcal{T}$ is freely generated over a set of base types, usually consisting of the types ι (for individuals) and o (for truth values), and the function type constructor $\rightarrow$):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha . s_\beta)_{\alpha\rightarrow\beta} \mid (s_{\alpha\rightarrow\beta}\, t_\alpha)_\beta \mid (\neg_{o\rightarrow o}\, s_o)_o \mid$$
$$(s_o \vee_{o\rightarrow o\rightarrow o}\, t_o)_o \mid (s_\alpha =_{\alpha\rightarrow\alpha\rightarrow o}\, t_\alpha)_o \mid (\Pi_{(\alpha\rightarrow o)\rightarrow o}\, s_{\alpha\rightarrow o})_o$$

p_α denotes typed constants and X_α typed variables (distinct from p_α). Complex typed terms are constructed via abstraction and application. Our logical connectives of choice are $\neg_{o\rightarrow o}$, $\vee_{o\rightarrow o\rightarrow o}$, $=_{\alpha\rightarrow\alpha\rightarrow o}$ and $\Pi_{(\alpha\rightarrow o)\rightarrow o}$ (for each type α).[6] From these connectives, other logical connectives can be defined in the usual way (e.g., $\wedge$ and $\Rightarrow$). We often use binder notation $\forall X_\alpha . s$ for $\Pi_{(\alpha\rightarrow o)\rightarrow o}(\lambda X_\alpha . s_o)$. We use the .-notation to avoid brackets; the convention is as follows: . stands for a pair of brackets whose right counterpart reaches as far to the right as is consistent with the logical structure and the type structure of an expression.

We assume familiarity with α-conversion, β- and η-reduction, and the existence of β- and $\beta\eta$-normal forms (see e.g. [66]). Moreover, we obey the usual definitions of free variable occurrences and substitutions.

The semantics of HOL is well understood and thoroughly documented in the literature [67, 68, 45, 69]. The semantics of choice for our work is Henkin semantics.

The encoding of HOL in THF0 syntax is straightforward. For example, `$i` and `$o` represent the standard base types ι and o, and `$i > $o` encodes a function (predicate) type. Function or predicate application as e.g. in the HOL formula (*loves ben mary*) is encoded in THF0 as `((loves @ Ben) @ Mary)` or simply as `(loves @ Ben @ Mary)`, that is, in THF0 we explicitly represent function application with the operator `@`. Universal and existential quantification and λ-abstraction as in $\forall X . \exists Y . (loves\ X\ Y)$ and $\lambda X . (loves\ ben\ X)$ are represented in THF0 as `![X:$i]: ?[Y:$i]: (loves @ X @ Y)` and `^[X:$i]: (loves @ Ben @ X)`; note that we have here explicitly assigned type `$i` to variables `X` and `Y`. The logical connectives $\neg, \vee, \wedge, \Rightarrow$ and $\Leftrightarrow$ are written as `~`, `|`, `&`, `=>`, and `<=>`.

THF0 encodings 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.

For further details on THF0 we refer to Sutcliffe and Benzmüller [9].

4.2. Translating SUMO to THF0

In our translation of SUMO to THF0 we recursively analyze all SUMO terms and subterms in order to assign consistent type

[5] There are other possible answers for ?*R*, including inequality and the universal relation. Enumerating them all and separating trivial, uninteresting answers from interesting ones is a challenge for future work.

[6] This choice is not minimal (from $=_{\alpha\rightarrow\alpha\rightarrow o}$ all other logical constants can already be defined [65]). It is useful though in the context of resolution based theorem proving.

information to them. In particular, we extract type information for all constant and variable symbols as required in THF0 files. For example, when applying our transformation procedure to P6.3 we generate the THF0 information given below.

```
%%% The extracted Signature %%%
thf(type_decl_1,type,(
    grandparent_IiioI: $i > $i > $o )).

thf(type_decl_2,type,(
    lCardinalityFn_IIioIiI: ( $i > $o ) > $i )).

thf(type_decl_3,type,(
    lJohn_i: $i )).

thf(type_decl_4,type,(
    ltet_IiioI: $i > $i > $o )).

thf(type_decl_5,type,(
    n3_i: $i )).

%%% The translated axioms %%%
thf(a1,axiom,(
    ltet_IiioI
    @ (lCardinalityFn_IIioIiI
       @ (^[X:$i]: (grandparent_IiioI
                    @ lJohn_i
                    @ X)))
    @ n3_i )).
```

Thereby we employ the following mapping of SUMO symbols to THF0 symbols: *grandparent* is mapped to `grandparent_IiioI`, and *CardinalityFn* and *John* to `lCardinalityFn_IIioIiI` and `lJohn_i`, respectively. These constant symbols are of types `$i > $i > $o`, `($i > $o) > $i` and `$i`. In our mapping we have chosen to represent type information as suffixes in the mapped names. For example, suffix `IIioIiI` in `lCardinalityFn_IIioIiI` encodes the type `($i > $o) > $i`; `I` is used for bracketing. This is in addition to the type declaration we anyway have to provide; the reason for doing this will become clear below. Moreover, THF0 constant symbols have to start with a lower case symbol which explains the leading *l*'s.

Some mappings of SUMO symbols are treated in a special way. For example, the arithmetic relation *lessThanOrEqualTo* and the number 3 are mapped to distinguished symbols `ltet_IiioI` and `n3_i`; the motivation thereby is to provide some special support for arithmetic reasoning in THF0 provers in the future.

The output of our SUMO to THF0 transformation is not intended for user consumption. It serves the main purpose of communicating SUMO reasoning problems to higher-order automated theorem provers and model finders.

So far, we use THF0 type `$i` as only base type other than `$o`. Hence, SUMO formulas are mapped to type `$o` while basic constants such as `lJohn_i` and `n3_i` are currently both declared of type `$i`. Function types, e.g. for `lCardinalityFn_IIioIiI`, are determined by our translation algorithm. The introduction of further base types (including, for example, a special type for naturals) in combination with a better exploitation of the richer 'type' information already available in SUMO should be straightforward, and future work should study which improvements are possible in THF0 reasoners when a richer type system is exploited.

Assigning types to SUMO terms is in fact not as straightforward as this small example suggests. A major problem is that SUMO supports self-applications as illustrated, for example, by the SUMO axiom

(*instance instance BinaryPredicate*)

In order to translate such axioms we currently split affected constants like *instance* in the THF0 mapping into separate constants, here we get `instance_IiioI` of type `$i > $i > $o` and `instance_IIiioIioI` of type `($i > $i > $o) > $i > $o`. This explains why we have chosen to include type information in the mapped symbol names.

```
%%% The extracted Signature %%%
thf(type_decl_1,type,(
    lBinaryPredicate_i: $i )).

thf(type_decl_2,type,(
    instance_IIiioIioI:
      ( $i > $i > $o ) > $i > $o )).

thf(type_decl_2,type,(
    instance_IiioI: $i > $i > $o )).

%%% The translated axiom(s) %%%
thf(a1,axiom,(
    instance_IIiioIioI
    @ instance_IiioI
    @ lBinaryPredicate_i )).
```

Obviously we may thereby lose relevant information. In our example we now only know for symbol `instance_IiioI` that it denotes a binary relation. If we want this information restored also for `instance_IIiioIioI` we can iterate the process and generate another symbol `instance_IIIiioIioIioI` and another axiom

```
thf(a2,axiom,(
    instance_IIIiioIioIioI
    @ instance_IIiioIioI
    @ lBinaryPredicate_i )).
```

Future work will study the practical need for such an iterated generation of axioms more closely. So far we have not come across practically motivated examples that do require it.

An important intermediate goal has thus been achieved, namely to provide a first translation of the SUMO upper ontology into THF0 that can be parsed and type checked by all THF0 reasoners in the TPTP. This THF0 translation of the SUMO upper ontology is available at: `http://christoph-benzmueller.de/papers/SUMO.thf`. SUMO documentation axioms are not relevant for the theorem provers and they have not been translated into THF0. This is why we obtain only 3577 THF0 axioms out of the approx. 4000 axioms in the original SUMO upper ontology.

5. Experiments

We have implemented the SUMO to THF0 translation algorithm as part of the Sigma ontology engineering environment. This enabled the reuse of already existing infrastructure, for example, for manipulating formulas and knowledge bases. Additionally, we have integrated the LEO-II system with Sigma.

There are now three modes in which LEO-II can be applied to queries in Sigma. The local mode only translates the user assertions and the query, the global mode translates the entire SUMO upper level ontology (resulting in the mentioned 3577 THF0 axioms) and then adds the user assertions and the query, and the SInE mode employs Hoder's SInE relevance filtering system [17] to extract a (hopefully) relevant subset of the axioms from the SUMO knowledge base, which is then translated into THF0.

We have conducted an initial experiment with the LEO-II prover (version v1.2.8). LEO-II provides an own relevance filtering mechanism, and in our experiment this relevance filtering was always enabled.[7] In the following we call this flag setting of LEO-II the SUMO setting.

LEO-II's relevance filtering mechanism is in fact still very basic. It is based on a symbol distance rating between the axioms and the given user query. The algorithm computes for each axiom A_i in the knowledge base the set $Consts(A_i)$ of constant symbols in A_i. A respective computation of contained constant symbols is also done for the formula set Q^1, which initially only contains the given user query; this set is called $Consts(Q^1)$. Next, we compute the set $Filtered^1 = \{A_i | Consts(A_i) \cap Consts(Q^1) \neq \emptyset\}$. The process can be iterated as follows. For $n > 1$ take $Filtered^n = Filtered^{n-1} \cup \{A_i | Consts(A_i) \cap Consts(Q^{n-1}) \neq \emptyset\}$, where Q^k is defined as $Q^{k-1} \cup Filtered^{k-1}$ for all $k > 1$. In the SUMO setting of LEO-II the iteration of this filtering mechanism is currently applied up to level $n = 2$.

In our experiment the maximum timeout for LEO-II was set to 300 seconds. All experiment runs were done on a standard iMac8,1 with a 2.8 GHz Intel Core 2 Duo processor and 2 GB of memory.

The results of this experiment are presented in Table 1. We report LEO-II's reasoning times in SUMO setting (in seconds) when solving the example problems as generated by Sigma in the respective modes. For practical use the SInE mode appears the most appropriate approach. However, even LEO-II alone is already capable of dealing with large knowledge bases as the results in the global mode confirm. This is due to LEO-II's own relevance filtering capabilities. In Examples 6 and 8 LEO-II finds a different proof in SInE mode than in local mode, which explains LEO-II's good performance for the SInE modes of these examples.

Several related example problems, including the ones from Table 1 and/or adaptations of them, have been added to the TPTP library. They are available under TPTP identifiers CSR119–CSR153.

[7] The exact command options for LEO-II employed in our experiments were: `leo <problem-file> -rf 2 -t 300`.

We have conducted a second series of experiments in which the higher-order reasoning systems TPS (version 3.110228S1a), Satallax (version 2.3), and Isabelle (version 2011) were applied in addition to LEO-II. These systems are all available online via the SystemOnTPTP tool [51]. Exploiting the TPTP World infrastructure [70], all experiment runs reported in Table 2 were done remotely at the University of Miami on 2.80GHz computers with 1GB memory and running the Linux operating system. The timeout in each run was set to 300 seconds. In this experiment LEO-II was employed in its standard setting as opposed to the SUMO setting. In the standard setting LEO-II first tackles a problem with relevance filtering disabled. Subsequently LEO-II then tries different reasoning modes some of which also have relevance filtering enabled.

Some interesting observations are:

- When LEO-II is applied in its standard setting, in which relevance filtering only gets enabled after some time, then then the reasoning times in the global mode of the experiments get significantly worse. That shows that relevance filtering is essential for LEO-II to solve these global mode problems on its own. However, in the SInE mode of the experiments the problem files were still small enough so that additional relevance filtering in LEO-II was not needed to obtain fast results.

- None of the other THF0 provers can solve the problems in global mode. Relevance filtering is obviously a missing feature in (the standard settings of) these systems.

- In SInE mode, where LEO-II still solves all of the problems effectively, only Satallax and Isabelle show some small successes.

- In local mode all our problems can be effectively solved not only by LEO-II but also by Satallax and TPS. Isabelle, however, performs weak and it surprisingly even fails on Problem 6 which it solves in the harder SInE mode.

Form these observations we conclude that LEO-II is currently the most promising reasoner for SUMO. If this picture should change in the future, then our flexible infrastructure supports an easy replacement of the prioritized THF0 reasoner in Sigma.

An overall conclusion from our experiments is that higher-order automated reasoners apparently can advance the automation of higher-order aspects in SUMO, provided they are applied with relevance filtering enabled (if available) or in combination with an additional relevance filtering systems such as SInE as preprocessor. We conjecture that this result is transferable to other expressive ontologies. However, much further work is needed to confirm this conjecture.

Note that first-order translation tricks such as employed, for example, by Pease and Sutcliffe [62] fail for the examples studied here, except probably for the trivial Examples 1 and 7. The added value of higher-order automated reasoning in SUMO has also been confirmed by the detection (and subsequent fixing) of some problematic axioms in the course of our experiments.

Mode	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5	Ex.6	Ex.7	Ex.8	Ex.9
local	0.112	0.109	0.116	0.082	0.079	0.277	0.143	0.080	0.063
SInE	0.309	0.294	0.396	0.213	0.363	0.124	0.394	0.071	0.111
global	3.818	3.791	3.320	3.246	3.189	2.590	4.522	2.820	2.532

Table 1: Performance of the LEO-II prover (in SUMO setting) for the examples problems in this article; three different problem modes were investigated.

Mode	System	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5	Ex.6	Ex.7	Ex.8	Ex.9
local	LEO-II	0.21	0.21	0.18	0.13	0.12	0.44	0.31	0.15	0.09
	TPS	4.46	4.47	8.64	8.65	6.65	8.69	0.39	0.41	21.45
	Satallax	0.01	0.01	1.13	0.21	2.37	14.81	0.01	0.01	0.03
	Isabelle	-	-	-	-	-	-	2.60	-	18.65
SInE	LEO-II	0.61	0.62	0.61	0.37	0.57	0.27	0.80	2.46	0.21
	TPS	-	-	-	-	-	-	-	-	-
	Satallax	-	0.12	-	-	-	3.46	0.03	0.62	-
	Isabelle	-	-	-	-	-	48.12	12.03	-	-
global	LEO-II	46.49	44.55	41.17	46.03	45.51	41.36	47.99	44.09	43.08
	TPS	-	-	-	-	-	-	-	-	-
	Satallax	-	-	-	-	-	-	6.77	87.42	-
	Isabelle	-	-	-	-	-	-	-	-	-

Table 2: Performance of the TPTP THF0 reasoners (in standard setting) for the example problems in this article; three different problem modes were investigated.

These problem-axioms have remained undetected by the incremental tests with first-order provers as reported in Section 2.1. 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)))
```

Most importantly, not only are various higher-order theorem provers now in principle applicable to SUMO but also the higher-order model finders Refute and Nitpick. For example, with their help it has been easy to detect typos in earlier modelings of our running examples.

6. A Proper Treatment of Modal Contexts in SUMO

We have illustrated in Section 3.4 that assuming Boolean extensionality for SUMO is in conflict with SUMO's modeling of epistemic and doxastic contexts. A solution to this problem is to model SUMO's modal operators as proper modalities in quantified multimodal logic (QML). That is, instead of translating SUMO directly into classical higher-order logic we now translate SUMO into QML. This enables the mapping of epistemic contexts like *(knows Peter <whatever>)* or doxastic contexts like *(believes Peter <whatever>)* to proper modalities in modal logic like $\Box_{KnowledgePeter}$ <*whatever*> and $\Box_{BelievesPeter}$ <*whatever*>. The need for quantifiers and for multiple modalities is obvious from our examples so far. We may add respective axioms in order to appropriately characterize the modalities we obtain and to specify their interaction. For example, to appropriately characterize $\Box_{KnowledgePeter}$ as an epistemic modality we may use the S5 axioms and to characterize $\Box_{BelievesPeter}$ as an doxastic modality we may use the S45 axioms. Moreover, an inclusion axiom between Peter's knowledge and Peter's beliefs can be added.

This approach connects SUMO's modeling of modal contexts with solid and well understood modelings of modalities as studied in the modal logics world. While this is theoretically interesting it raises concerns regarding its practicality. The challenge clearly is to provide powerful practical reasoning systems for QML. In particular, these systems would need to support varying combinations of modal operators. Unfortunately, however, there are currently only very few specialist reasoners available for quantified monomodal logics. The available reasoners include MleanTAP and MleanSeP [71], GQML [72], and f2p+MSPASS (which is an extension of the MSPASS prover [73]). To the best of our knowledge, none of these systems currently supports flexible combinations of different modalities as required for SUMO. Moreover, these systems are generally restricted to first-order quantification only, so that they cannot (easily) address other higher-order aspects as mentioned in Section 3.

For this reason we have developed an alternative automation approach for QML. This approach exploits our recent semantic embedding of QML in HOL [74]. This embedding demonstrates that QML is actually just a natural fragment of HOL respectively THF0.

Hence, instead of mapping SUMO directly to THF0 as in

Section 4, we now take a detour via QML. In the end we nevertheless obtain a proper THF0 encoding, which enables the application of off-the-shelf higher-order reasoners such as our LEO-II prover. An advantage of this approach is its flexibility, since arbitrary numbers and combinations of (not only) epistemic and doxastic modalities are supported. Moreover, the approach still scales to other higher-order aspects in SUMO.

A recent case study by Otten and Raths on automating quantified monomodal logics [71] actually confirms that higher-order automated theorem provers such as Satallax and LEO-II can in fact compete with the above mentioned specialist reasoners for reasoning in quantified monomodal logics when using our semantic embeddings based approach. Moreover, a number of alternative examples requiring combinations of modalities have been studied and automated [75]. These examples together with their performance results are available in the TPTP THF library (cf. [75] for further details) and they provide first evidence for the practicality of our proposed solution.

6.1. Embedding QML in THF0

Quantified modal logics have been studied by Fitting [76] (further related work is available by Blackburn and Marx [77] and Braüner [78]). In contrast to Fitting we are here not interested only in **S5** structures but in the more general case of **K** from which more constrained structures (such as **S5**) can be easily obtained by adding axioms. First-order quantification can be constant domain or varying domain. Below we only consider the constant domain case: every possible world has the same domain. While Fitting [76] studies quantified monomodal logic, we are interested in multiple modalities. Hence, we introduce multiple $\Box_r$ operators for symbols r from an index set S. The grammar for our quantified multimodal logic QML thus is

$$s, t ::= P \mid k(X^1, \ldots, X^n) \mid \neg\, s \mid s \vee t \mid \forall X_{\cdot}\, s \mid \forall P_{\cdot}\, s \mid \Box_r\, s$$

where $P \in \mathrm{PV}$ denotes propositional variables, $X, X^i \in \mathrm{IV}$ denote first-order (individual) variables, and $k \in \mathrm{SYM}$ denotes predicate symbols of any arity ($n \geq 0$). Further connectives, quantifiers, and modal operators can be defined as usual. We also obey the usual definitions of free variable occurrences and substitutions.

Fitting introduces three different notions of Kripke semantics for QML: **QS5**π^-, **QS5**π, and **QS5**π^+. In our work [74, 79] we study related notions **QK**π^-, **QK**π, and **QK**π^+ for a modal context **K**, and we support multiple modalities.

HOL is an expressive logic and it is thus not surprising that QML can be elegantly modeled and even automated as a fragment of HOL. The idea of the encoding, called $\mathrm{QML}^{\mathrm{HOL}}$, is simple. Choose type ι to denote the (non-empty) set of individuals and choose an additional base type μ to denote the (non-empty) set of possible worlds. As usual, the type o denotes the set of truth values. Certain formulas of type $\mu \to o$ then correspond to multimodal logic expressions. The multimodal connectives $\neg$, $\vee$, and $\Box$, become λ-terms of types $(\mu \to o) \to (\mu \to o)$, $(\mu \to o) \to (\mu \to o) \to (\mu \to o)$, and $(\mu \to \mu \to o) \to (\mu \to o) \to (\mu \to o)$ respectively.

Quantification is handled as in HOL by modeling $\forall X_{\cdot}\, p$ as $\Pi(\lambda X_{\cdot}\, p)$ for a suitably chosen connective Π. Here we are interested in defining two particular modal $\mathbf{\Pi}$-connectives: $\mathbf{\Pi}^{\iota}$, for quantification over individual variables, and $\mathbf{\Pi}^{\mu\to o}$, for quantification over modal propositional variables that depend on worlds. They become terms of type $(\iota \to (\mu \to o)) \to (\mu \to o)$ and $((\mu \to o) \to (\mu \to o)) \to (\mu \to o)$ respectively.

The $\mathrm{QML}^{\mathrm{HOL}}$ modal operators $\neg, \vee, \Box, \mathbf{\Pi}^{\iota}$, and $\mathbf{\Pi}^{\mu\to o}$ are now simply defined as follows:

$$\begin{aligned}
\neg_{(\mu\to o)\to(\mu\to o)} &= \lambda\phi_{\mu\to o\cdot}\, \lambda W_{\mu\cdot}\, \neg\phi\, W \\
\vee_{(\mu\to o)\to(\mu\to o)\to(\mu\to o)} &= \lambda\phi_{\mu\to o\cdot}\, \lambda\psi_{\mu\to o\cdot}\, \lambda W_{\mu\cdot}\, \phi\, W \vee \psi\, W \\
\Box_{(\mu\to\mu\to o)\to(\mu\to o)\to(\mu\to o)} &= \lambda R_{\mu\to\mu\to o\cdot}\, \lambda\phi_{\mu\to o\cdot} \\
&\qquad \lambda W_{\mu\cdot}\, \forall V_{\mu\cdot}\, \neg R\, W\, V \vee \phi\, V \\
\mathbf{\Pi}^{\iota}_{(\iota\to(\mu\to o))\to(\mu\to o)} &= \lambda\phi_{\iota\to(\mu\to o)\cdot}\, \lambda W_{\mu\cdot}\, \forall X_{\iota\cdot}\, \phi\, X\, W \\
\mathbf{\Pi}^{\mu\to o}_{((\mu\to o)\to(\mu\to o))\to(\mu\to o)} &= \lambda\phi_{(\mu\to o)\to(\mu\to o)\cdot}\, \lambda W_{\mu\cdot}\, \forall P_{\mu\to o\cdot}\, \phi\, P\, W
\end{aligned}$$

Note that this encoding actually only employs the second-order fragment of HOL enhanced with lambda-abstraction. However, if we decide to include further $\mathbf{\Pi}$ operators for higher types (which is straightforward to do [80]) then the second-order fragment of HOL is not sufficient anymore.

Further modal operators can be introduced as usual, for example, $\top = \lambda W_{\mu\cdot}\, \top$, $\bot = \neg\top$, $\wedge = \lambda\phi, \psi_{\cdot}\, \neg(\neg\phi \vee \neg\psi)$, $\supset = \lambda\phi, \psi_{\cdot}\, \neg\phi \vee \psi$, $\Leftrightarrow = \lambda\phi, \psi_{\cdot}\, (\phi \supset \psi) \wedge (\psi \supset \phi)$, $\Diamond = \lambda R, \phi_{\cdot}\, \neg(\Box\, R\, (\neg\phi))$, $\mathbf{\Sigma}^{\iota} = \lambda\phi_{\cdot}\, \neg\, \mathbf{\Pi}^{\iota}(\lambda X_{\cdot}\, \neg\, \phi\, X)$, $\mathbf{\Sigma}^{\mu\to o} = \lambda\phi_{\cdot}\, \neg\, \mathbf{\Pi}^{\mu\to o}(\lambda P_{\cdot}\, \neg\, \phi\, P)$.

For defining $\mathrm{QML}^{\mathrm{HOL}}$ propositions we fix a set $\mathrm{IV}^{\mathrm{HOL}}$ of individual variables of type ι, a set $\mathrm{PV}^{\mathrm{HOL}}$ of propositional variables[8] of type $\mu \to o$, and a set $\mathrm{SYM}^{\mathrm{HOL}}$ of n-ary (curried) predicate symbols of types $\underbrace{\iota \to \ldots \to \iota}_{n} \to (\mu \to o)$. Moreover, we fix a set $\mathrm{S}^{\mathrm{HOL}}$ of accessibility relation constants of type $\mu \to \mu \to o$. $\mathrm{QML}^{\mathrm{HOL}}$ propositions are now defined as the smallest set of HOL-terms for which the following hold:

- if $P \in \mathrm{PV}^{\mathrm{HOL}}$, then $P \in \mathrm{QML}^{\mathrm{HOL}}$
- if $X^j \in \mathrm{IV}^{\mathrm{HOL}}$ ($j = 1, \ldots, n$; $n \geq 0$) and $k \in \mathrm{SYM}^{\mathrm{HOL}}$, then $(k\, X^1\ \ldots\ X^n) \in \mathrm{QML}^{\mathrm{HOL}}$
- if $\phi, \psi \in \mathrm{QML}^{\mathrm{HOL}}$, then $\neg\, \phi \in \mathrm{QML}^{\mathrm{HOL}}$ and $\phi \vee \psi \in \mathrm{QML}^{\mathrm{HOL}}$
- if $r \in \mathrm{S}^{\mathrm{HOL}}$ and $\phi \in \mathrm{QML}^{\mathrm{HOL}}$, then $\Box\, r\, \phi \in \mathrm{QML}^{\mathrm{HOL}}$
- if $X \in \mathrm{IV}^{\mathrm{HOL}}$ and $\phi \in \mathrm{QML}^{\mathrm{HOL}}$, then $\mathbf{\Pi}^{\iota}(\lambda X_{\cdot}\, \phi) \in \mathrm{QML}^{\mathrm{HOL}}$
- if $P \in \mathrm{PV}^{\mathrm{HOL}}$ and $\phi \in \mathrm{QML}^{\mathrm{HOL}}$, then $\mathbf{\Pi}^{\mu\to o}(\lambda P_{\cdot}\, \phi) \in \mathrm{QML}^{\mathrm{HOL}}$

We write $\Box_r\, \phi$ for $\Box\, r\, \phi$, $\forall X_{\iota\cdot}\, \phi$ for $\mathbf{\Pi}^{\iota}(\lambda X_{\iota\cdot}\, \phi)$, $\forall P_{\mu\to o\cdot}\, \phi$ for $\mathbf{\Pi}^{\mu\to o}(\lambda P_{\mu\to o\cdot}\, \phi)$, $\exists X_{\iota\cdot}\, \phi$ for $\neg\, \mathbf{\Pi}^{\iota}(\lambda X_{\iota\cdot}\, \neg\, \phi)$, and $\exists P_{\mu\to o\cdot}\, \phi$ for $\neg\, \mathbf{\Pi}^{\mu\to o}(\lambda P_{\mu\to o\cdot}\, \neg\, \phi)$,

If type information is obvious we may avoid displaying it.

[8] Note that the denotation of propositional variables depends on worlds.

Note that the defining equations for our QML modal operators are themselves formulas in HOL. Hence, we can express QML formulas in a higher-order prover elegantly in the usual syntax (and the theorem prover may subsequently expand the definitions of the contained modal operators). For example,

$$\Box_r \, \exists P_{\mu\to o}.\ P$$

is a QML$^{\text{HOL}}$ proposition; it has type $\mu \to o$.

Validity of QML$^{\text{HOL}}$ propositions is defined in the obvious way: a QML$^{\text{HOL}}$ proposition $\phi_{\mu\to o}$ is valid if and only if for all possible worlds w_μ we have $w \in \phi_{\mu\to o}$, that is, if and only if $\phi_{\mu\to o}\, w_\mu$ holds. Hence, the notion of validity is modeled via the following equation (alternatively we could define *vld* simply as $\Pi_{(\mu\to o)\to o}$):

$$vld = \lambda\phi_{\mu\to o}.\forall W_\mu.\phi\, W$$

Now we can formulate proof problems in QML$^{\text{HOL}}$, e.g.,

$$vld\ \Box_r \, \exists P_{\mu\to o}.\ P$$

By rewriting the definitions we can reduce such proof problems to corresponding statements containing only the basic connectives $\neg$, $\vee$, $=$, Π^ι, and $\Pi^{\mu\to o}$ of HOL. In contrast to the many other approaches no external transformation mechanism is required. For our example formula $vld\ \Box_r \, \exists P_{\mu\to o}.\ P$ unfolding and $\beta\eta$-reduction leads to

$$\forall W_\mu.\forall Y_\mu.\neg r\, W\, Y \vee (\neg\forall X_{\mu\to o}.\neg(X\, Y))$$

It is easy to check that this formula is valid in Henkin semantics: put $X = \lambda Y_\mu.\top$.

We have proved soundness and completeness for this embedding [74, 79].

The THF0 encoding of our embedding of quantified multimodal logic in HOL is available for inspection and easy reuse in the TPTP library under identifier `LCL013^0.ax`.

6.2. Mapping SUMO via QML to THF0

Exploiting the above embedding of quantified multimodal logic in HOL we can now suitably map SUMO problems via quantified multimodal logics to THFO. We illustrate the approach with an example. Local premises such as

```
(knows
   Chris
   (forall (?X)
      (=>
         (likes Mary ?X)
         (likes Sue ?X))))
(likes Mary Bill)
```

and SUMO ontology axioms such as

```
(=>
   (knows ?AGENT ?FORMULA)
   (truth ?FORMULA True))
```

are first lifted to respective QML$^{\text{HOL}}$ terms. For these example formulas we obtain (note that *likes* is now of type $\iota \to \iota \to \mu \to o$)

$$(\Box_{KnowsChris}\ \forall X_\iota.((likes\ Mary\ X)\ \supset\ (likes\ Sue\ X))) \quad (1)$$

$$(likes\ Mary\ Bill) \quad (2)$$

$$\forall A_{\iota\to\iota\to o}, F_{\mu\to o}.\ ((\Box_A\ F)\ \supset\ (truth\ F\ \top)) \quad (3)$$

These terms are all of type $\mu \to o$, that is, they are applicable to possible worlds. Subsequently, we have to ground these lifted terms to type o. To do so, terms related to T-Box like information (axioms) in SUMO, such as (3), are interpreted as universal for all possible worlds:

$$\forall W_\mu.((\forall A_{\iota\to\iota\to o}, F_{\mu\to o}.\ (\Box_A\ F)\ \supset\ (truth\ F\ \top))\ W)$$

which is equivalent to

$$vld\ (\forall A_{\iota\to\iota\to o}, F_{\mu\to o}.\ (\Box_A\ F)\ \supset\ (truth\ F\ \top))$$

A-Box like information such as our local premises (1) and (2) and queries are modeled with respect to a current world *cw* of type μ. For example, the query

$$(knows\ Chris\ (likes\ Sue\ Bill))$$

is mapped to

$$((\Box_{KnowsChris}\ (likes\ Sue\ Bill))\ cw)$$

Moreover, appropriate axioms are generated and added for each epistemic and doxastic modal operator. For example, for the epistemic modality $\Box_{KnowsChris}$ the following S5 axioms are added:

$$vld\ (\forall\phi_{\mu\to o}.\Box_{KnowsChris}\,\phi\ \supset\ \phi)$$
$$vld\ (\forall\phi_{\mu\to o}.\Diamond_{KnowsChris}\,\phi\ \supset\ \Box_{KnowsChris}\,\Diamond_{KnowsChris}\,\phi)$$

The disputed and rejected Example 8 is mapped in our modified translation approach to the following quantified multimodal logic encoding.[9]

Example 10 (Example 8 mapped to QML$^{\text{HOL}}$)**.**

$$\forall Y_{\mu\to\mu\to o}.((\Box_Y\ \top)\ cw) \quad \text{(P10.1)}$$

$$((likes\ Mary\ Bill)\ cw) \quad \text{(P10.2)}$$

$$((\Box_{KnowsChris}\ (\forall X_\iota.((likes\ Mary\ X)\ \supset\ (likes\ Sue\ X))))\ cw) \quad \text{(P10.3)}$$

$$vld\ (\forall\phi_{\mu\to o}.\Box_{KnowsChris}\,\phi\ \supset\ \phi) \quad \text{(P10.4)}$$

$$vld\ (\forall\phi_{\mu\to o}.\Diamond_{KnowsChris}\,\phi\ \supset\ \Box_{KnowsChris}\,\Diamond_{KnowsChris}\,\phi) \quad \text{(P10.5)}$$

$$((\Box_{KnowsChris}\ (likes\ Sue\ Bill))\ cw) \quad \text{(Q10)}$$

[9] It does not make a difference whether we use tautology $\top$ or (*equal Chris Chris*) in Premise P10.1

Exploiting our QML^{HOL} embedding we can thus obtain a proper THF0 problem encoding for Example 10 and we can hence apply higher-order automated reasoners to it.

Example 10 is not valid anymore, which is what we wanted to achieve. LEO-II fails to prove the query (within a 24 hours timeout). However, when Premise P10.2 is moved into the context of Chris' knowledge, then we get the following modified situation:

Example 11 (Modified Example 10)**.**

$$\forall Y_{\mu\to\mu\to o}.((\Box_Y\ \top)\ cw) \quad \text{(P11.1)}$$

$$((\Box_{KnowsChris}\ (likes\ Mary\ Bill))\ cw) \quad \text{(P11.2)}$$

$$((\Box_{KnowsChris}\ (\forall X_\iota.((likes\ Mary\ X)\supset(likes\ Sue\ X))))\ cw) \quad \text{(P11.3)}$$

$$vld\ (\forall\phi_{\mu\to o}.\Box_{KnowsChris}\ \phi\supset\phi) \quad \text{(P11.4)}$$

$$vld\ (\forall\phi_{\mu\to o}.\Diamond_{KnowsChris}\ \phi\supset\Box_{KnowsChris}\Diamond_{KnowsChris}\ \phi) \quad \text{(P11.5)}$$

——

$$((\Box_{KnowsChris}\ (likes\ Sue\ Bill))\ cw) \quad \text{(Q11.6)}$$

In this modified situation the query is valid (also without P11.4 and P11.5) and it is proved by LEO-II in a fraction of a second. (A closer look at LEO-II's proof protocol reveals that premise P11.1 is not used in the proof, which is what we expected.)

The extension of our translation to other modal operators in SUMO besides *knows* and *believes* is straightforward. Moreover, there is already some evidence that our automation approach scales to at least some reasonable number of combinations and nestings of modal operators [75]. Since there is currently no practical system in the direct or first-order approach available that supports flexible combinations and nestings of modalities, a solid comparison with alternative approaches is not feasible at this stage.

7. Related Work

The study of notions of context has a long history in philosophy, linguistics, and artificial intelligence. In artificial intelligence, a main motivation has been to resolve the problem of generality of computer programs as identified by McCarthy [4]. Giunchiglia [6] additionally emphasizes locality and the need for structured representations of knowledge. Different approaches to formalizing context have been proposed in the last decades and they have been discussed in overview articles [81, 82].

McCarthy [5] pioneered the modeling of contexts as first class objects and he introduced the predicate *ist*. For example, in his approach the query *Q8* would be encoded as *ist(context_of("Chris's Knowledge"),likes(Sue,Bill))*. A main motivation of McCarthy's approach actually is to avoid modal logics and, moreover, to support rich and structured context descriptions. His line of research has been followed by a number of researchers, including, for example, Guha (who has put contexts into Cyc), Buvac, and Mason [83, 84]. Also Giunchiglia and Serafini [85] avoid modal logics and propose the use of so called multilanguage systems. They show various equivalence results to common modal logics, but they also discuss several properties of multilanguage systems not supported in modal logics.

All of the above approaches avoid a higher-order perspective on context. This is the main difference to the work presented here. However, we argue that a solid higher-order perspective on context can be very valuable for various reasons. On the theory side the twist between formalisms based on modal logic and formalisms based on first-order logic seems to dissolve, since both modal logics and first-order logics are just natural fragments of classical higher-order logics. Most importantly, encodings based on modal logics and first-order logic can be elegantly combined in classical higher-order logics. It is this representational power which we have exploited in our recent work [75, 79, 74, 86, 87] and which we do also employ here. Moreover, deeper semantic issues can be clarified when taking a solid higher-order perspective. On the practical side there are now several automated higher-order reasoning systems available in our approach that can be uniformly applied to (intuitively sound) formalizations of context.

This article combines and extends previous results [88, 89]. A main extension is that relevance filtering and scalability to large knowledge bases, which was still mentioned as future work before, has been included in the studies in this article. Relevance filtering can be seen as our means to address the locality criterion. Here we have provided evidence that well known relevance filtering techniques can be appropriately adapted to HOL and be combined with our approach to context reasoning in SUMO. Regarding practical relevance of our entire approach this evidence has been regarded as a crucial missing cornerstone by reviewers of the earlier work.

8. Conclusion

The work presented in this article initially had a very simple and practical motivation, namely to provide better automated reasoning support for SUMO problems containing temporal, epistemic and doxastic contexts. However, our overall findings are of much wider relevance:

- The Boolean extensionality conflict applies to SUMO operators other than *knows* and *believes*. Consider, for example, the operator *hasPurpose* and let us assume that the following axioms are given:

  (*hasPurpose*
  MedicationForBen
  (*recoversFromIllness Ben*))

  (*recoversFromIllness Ben*)

  (*recoversFromIllness Bill*)

  Boolean extensionality allows us to infer the intuitively

unsound statement

(*hasPurpose*
MedicationForBen
(*recoversFromIllness Bill*))

In SUMO further operators like *hasPurpose* may be introduced in user defined domain ontologies. Hence, an intuitively sound reasoning support is required in which each such operator can be flexibly assigned with an appropriate semantics. The HOL based approach sketched in this paper is well suited for this purpose.

- The Boolean extensionality conflict analogously applies to a range of related knowledge representation frameworks. Most prominently, it applies to McCarthy's *ist*-operator for which it analogously enables counterintuitive inferences. For example, using Boolean extensionality we can infer from

  ist(*context_of*(*"KnowledgeOfChris"*),
  1 + 1 = 2)

  that also

  ist(*context_of*(*"KnowledgeOfChris"*),
  <*FermatsLastTheorem*>)

  holds (where <*FermatsLastTheorem*> abbreviates a respective longer formula expression). To enable this inference all we need to know is that 1 + 1 = 2 and <*FermatsLastTheorem*> are both valid (and hence equivalent and hence equal).

  Our findings may also apply to frameworks that evolved from McCarthy's pioneering work, including the language CycL.[10]

Future work includes the combination of temporal, epistemic and doxastic contexts as discussed in this article with further kinds of contexts and other challenge aspects in SUMO. This line of research will adapt, extend and exploit our recent embeddings of intuitionistic logics [74], logics for spatial reasoning [75], conditional logics [87, 90] and logics for security [86] in HOL. Further case studies are required to determine the scalability of the presented approach for flexible combinations and nestings of contexts. Moreover, the scalability of relevance filtering for knowledge bases larger than SUMO needs further investigation. However, based on the evidence provided in this article we conjecture that recent improvements of relevance filtering techniques in first-order provers [91, 92, 18] can be easily adopted for the higher-order case.

The transferability of our approach to other expressive ontologies, such as Cyc or DOLCE, needs further investigation. For this we have already started to encode the DOLCE ontology in THF0.[11]

Future work could also try to systematically reconstruct and embed several of the most prominent other notions of context. Such a project should in particular re-investigate the simple type theory based proposals by Church [93, 94, 95], Montague [96], and Thomason [97].

Acknowledgment:. We would like to thank Geoff Sutcliffe and Larry Paulson for their support and their contributions to our work. Moreover, we thank the reviewers of this article for many fruitful comments.

References

[1] A. Pease (Ed.), Ontology: A Practical Guide, Articulate Software Press, Angwin, CA 94508, 2011.

[2] I. Niles, A. Pease, Towards a standard upper ontology, in: FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems, ACM, New York, NY, USA, 2001, pp. 2–9.

[3] D. Ramachandran, P. Reagan, K. Goolsbey, First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology, in: S. P. (Ed.), 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, 2005.

[4] J. McCarthy, Generality in artificial intelligence, Communications of the ACM 30 (12) (1987) 1030–1035.

[5] J. McCarthy, Notes on formalizing context, in: Proceedings of IJCAI'93, 1993, pp. 555–562.

[6] F. Giunchiglia, Contextual reasoning, Epistemologia (Special Issue on Languages and Machines) 16 (1993) 345–364.

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

[8] C. Benzmüller, F. Rabe, G. Sutcliffe, THF0 - The Core TPTP Language for Classical Higher-Order Logic, in: P. Baumgartner, A. Armando, D. Gilles (Eds.), Proceedings of the 4th International Joint Conference on Automated Reasoning, no. 5195 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 2008, pp. 491–506.

[9] G. Sutcliffe, C. Benzmüller, Automated reasoning in higher-order logic using the TPTP THF infrastructure, Journal of Formalized Reasoning 3 (1) (2010) 1–27.

[10] C. Benzmüller, F. Theiss, L. Paulson, A. Fietzke, LEO-II — A cooperative automatic theorem prover for higher-order logic, in: A. Armando, P. Baumgartner, G. Dowek (Eds.), Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings, Vol. 5195 of Lecture Notes in Artificial Intelligence, Springer, 2008, pp. 162–170.

[11] I. Niles, A. Pease, Linking lexicons and ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, in: H. R. Arabnia (Ed.), Proceedings of the International Conference on Information and Knowledge Engineering. IKE'03, June 23 - 26, 2003, Las Vegas, Nevada, USA, Volume 2, CSREA Press, 2003, pp. 412–416.

[12] G. de Melo, F. Suchanek, A. Pease, Integrating YAGO into the Suggested Upper Merged Ontology, in: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), IEEE Computer Society, Los Alamitos, CA, USA, 2008.

[13] M. R. Genesereth, Knowledge interchange format, in: J. Allen, R. Fikes, E. Sandewall (Eds.), Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1991, pp. 238–249.

[14] A. Pease, The Sigma ontology development environment., in: F. Giunchiglia, A. Gomez-Perez, A. Pease, H. Stuckenschmid, Y. Sure, S. Willmott (Eds.), Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems, Vol. 71, CEUR Workshop Proceedings, 2003.

[10] `http://en.wikipedia.org/wiki/CycL`; the website says that CycL considers 5 different truth values, but a document with a clear semantics of CycL and in particular with a precise semantics of embedded formulas could not be found by the authors.

[11] `http://christoph-benzmueller.de/papers/DOLCE.thf`

[15] A. Riazanov, A. Voronkov, The Design and Implementation of Vampire, AI Communications 15 (2-3) (2002) 91–110.
[16] S. Trac, G. Sutcliffe, A. Pease, Integration of the TPTPWorld into SigmaKEE, in: B. Konev, R. Schmidt, S. Schulz (Eds.), Proceedings of the First International Workshop on Practical Aspects of Automated Reasoning (PAAR), Vol. 373 of CEUR Workshop Proceedings, CEUR-WS.org, 2008.
[17] K. Hoder, Automated reasoning in large knowledge bases, Master's thesis, Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague (2008).
[18] A. Pease, G. Sutcliffe, N. Siegel, S. Trac, Large theory reasoning with SUMO at CASC, AI Communications 23 (2-3) (2010) 137–144.
[19] S. Schulz, E: A Brainiac Theorem Prover, AI Communications 15 (2-3) (2002) 111–126.
[20] F. Frege, Grundgesetze der Arithmetik, Jena, 1893,1903.
[21] A. Church, A Formulation of the Simple Theory of Types, Journal of Symbolic Logic 5 (1940) 5668.
[22] J. Girard, The System F of Variable Types, Fifteen Years Later, Theoretical Computer Science 45 (2) (1986) 159–192.
[23] J. Reynolds, Types, Abstraction, and Parametric Polymorphism, no. 83 in Information Processing, North-Holland, 1986, pp. 513–523.
[24] N. de Bruijn, The Mathematical Language Automath, its Usage, and Some of its Extensions, in: R. Nederpelt, J. Geuvers, R. de Vrijer (Eds.), Selected Papers on Automath, no. 133 in Studies in Logic and the Foundations of Mathematics, North-Holland, 1994, pp. 73–100.
[25] P. Martin-Löf, Intuitionistic Type Theory, Bibliopolis, 1984.
[26] M. Gordon, T. Melham, Introduction to HOL, a Theorem Proving Environment for Higher Order Logic, Cambridge University Press, 1993.
[27] J. Harrison, HOL Light: A Tutorial Introduction, in: M. Srivas, A. Camilleri (Eds.), Proceedings of the 1st International Conference on Formal Methods in Computer-Aided Design, no. 1166 in Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 265–269.
[28] S. Owre, S. Rajan, J. Rushby, N. Shankar, M. Srivas, PVS: Combining Specification, Proof Checking, and Model Checking, in: R. Alur, T. Henzinger (Eds.), Computer-Aided Verification, no. 1102 in Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 411–414.
[29] T. Nipkow, L. Paulson, M. Wenzel, Isabelle/HOL: A Proof Assistant for Higher-Order Logic, no. 2283 in Lecture Notes in Computer Science, Springer-Verlag, 2002.
[30] J. Siekmann, C. Benzmüller, S. Autexier, Computer Supported Mathematics with OMEGA, Journal of Applied Logic 4 (4) (2006) 533–559.
[31] R. Nederpelt, J. Geuvers, R. de Vrijer, Selected Papers on Automath, no. 133 in Studies in Logic and the Foundations of Mathematics, North-Holland, 1994.
[32] S. Allen, M. Bickford, R. Constable, R. Eaton, C. Kreitz, L. Lorigo, E. Moran, Innovations in Computational Type Theory using Nuprl, Journal of Applied Logic 4 (4) (2006) 428–469.
[33] R. Pollack, The Theory of LEGO: A Proof Checker for the Extended Calculus of Constructions, Ph.D. thesis, University of Edinburgh, Edinburgh, United Kingdom (1994).
[34] A. Asperti, C. Sacerdoti Coen, E. Tassi, S. Zacchiroli, User Interaction with the Matita Proof Assistant, Journal of Automated Reasoning 39 (2) (2007) 109–139.
[35] Y. Bertot, P. Castéran, Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions, Texts in Theoretical Computer Science, Springer-Verlag, 2004.
[36] T. Coquand, G. Huet, The Calculus of Constructions, Information and Computation 76 (2-3) (1988) 95–120.
[37] F. Pfenning, Logic Programming in the LF Logical Framework, in: G. Huet, G. Plotkin (Eds.), Logical Frameworks, Cambridge University Press, 1991, pp. 149–181.
[38] F. Pfenning, C. Schürmann, System Description: Twelf - A Meta-Logical Framework for Deductive Systems, in: H. Ganzinger (Ed.), Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Proceedings, no. 1632 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1999, pp. 202–206.
[39] P. B. Andrews, Resolution in Type Theory, Journal of Symbolic Logic 36 (3) (1971) 414–432.
[40] G. Huet, A Unification Algorithm for Typed Lambda-Calculus, Theoretical Computer Science 1 (1) (1975) 27–57.
[41] G. Huet, A Complete Mechanization of Type Theory, in: Proceedings of the 3rd International Joint Conference on Artificial Intelligence , 1973, pp. 139–146.
[42] T. Pietrzykowski, D. Jensen, A Complete Mechanization of Omega-order Type Theory, in: J. Donovan, R. Shields (Eds.), Proceedings of the ACM Annual Conference, ACM Press, 1972, pp. 82–92.
[43] C. Benzmüller, M. Kohlhase, Extensional higher-order resolution, in: C. Kirchner, H. Kirchner (Eds.), Automated Deduction - CADE-15, 15th International Conference on Automated Deduction, Proceedings, no. 1421 in Lecture Notes in Artificial Intelligence, Springer, 1998, pp. 56–71.
[44] C. Benzmüller, Extensional higher-order paramodulation and RUE-resolution, in: H. Ganzinger (Ed.), Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Proceedings, no. 1632 in Lecture Notes in Artificial Intelligence, Springer, 1999, pp. 399–413.
[45] C. Benzmüller, C. Brown, M. Kohlhase, Higher-order semantics and extensionality, Journal of Symbolic Logic 69 (4) (2004) 1027–1088.
[46] C. E. Brown, Automated Reasoning in Higher-Order Logic: Set Comprehension and Extensionality in Church's Type Theory, no. 10 in Studies in Logic: Logic and Cognitive Systems, College Publications, 2007.
[47] P. B. Andrews, M. Bishop, S. Issar, N. D., F. Pfenning, H. Xi, TPS: A Theorem-Proving System for Classical Type Theory, Journal of Automated Reasoning 16 (3) (1996) 321–353.
[48] P. B. Andrews, C. E. Brown, TPS: A hybrid automatic-interactive system for developing proofs, Journal of Applied Logic 4 (4) (2006) 367–395.
[49] G. Sutcliffe, The TPTP problem library and associated infrastructure, Journal of Automated Reasoning 43 (4) (2009) 337–362.
[50] G. Sutcliffe, C. Benzmüller, C. Brown, F. Theiss, Progress in the development of automated theorem proving for higher-order logic, in: R. Schmidt (Ed.), Automated Deduction - CADE-22, 22th International Conference on Automated Deduction, Proceedings, Vol. 5663 of Lecture Notes in Artificial Intelligence, Springer, 2009, pp. 116–130.
[51] G. Sutcliffe, TPTP, TSTP, CASC, etc., in: V. Diekert, M. Volkov, A. Voronkov (Eds.), Proceedings of the 2nd International Computer Science Symposium in Russia, no. 4649 in Lecture Notes in Computer Science, Springer-Verlag, 2007, pp. 7–23.
[52] C. Benzmüller, M. Kohlhase, LEO – a higher-order theorem prover, in: C. Kirchner, H. Kirchner (Eds.), Automated Deduction - CADE-15, 15th International Conference on Automated Deduction, Proceedings, no. 1421 in Lecture Notes in Artificial Intelligence, Springer, Lindau, Germany, 1998, pp. 139–143.
[53] C. Benzmüller, V. Sorge, M. Jamnik, M. Kerber, Combined reasoning by automated cooperation, Journal of Applied Logic 6 (2008) 318–342.
[54] S. Schulz, E – A Brainiac Theorem Prover, Journal of AI Communications 15 (2/3) (2002) 111–126.
[55] J. Hurd, First-Order Proof Tactics in Higher-Order Logic Theorem Provers, in: M. Archer, B. Di Vito, C. Munoz (Eds.), Proceedings of the 1st International Workshop on Design and Application of Strategies/Tactics in Higher Order Logics, no. NASA/CP-2003-212448 in NASA Technical Reports, 2003, pp. 56–68.
[56] D. Miller, A Compact Representation of Proofs, Studia Logica 46 (4) (1987) 347–370.
[57] P. B. Andrews, Theorem Proving via General Matings, Journal of the ACM 28 (2) (1981) 193–214.
[58] C. E. Brown, Reducing higher-order theorem proving to a sequence of sat problems, in: N. Bjorner, V. Sofronie-Stockkermans (Eds.), CADE the 23rd International Conference on Automated Deduction, LNCS/LNAI 6803, Springer, 2011, pp. 147 – 161.
[59] J. Backes, C. E. Brown, Analytic tableaux for higher-order logic with choice, in: J. Giesl, R. Hähnle (Eds.), Automated Reasoning: 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010, Proceedings, Vol. 6173 of Lecture Notes in Artificial Intelligence, Springer, 2010, pp. 76–90.
[60] T. Weber, Sat-based finite model generation for higher-order logic, Ph.D. thesis, Dept. of Informatics, T.U. München (2008).
[61] J. C. Blanchette, T. Nipkow, Nitpick: A counterexample generator for higher-order logic based on a relational model finder, in: M. Kaufmann, L. C. Paulson (Eds.), Proceedings of ITP 2010, Vol. 6172 of Lecture Notes in Computer Science, Springer, 2010, pp. 131–146.
[62] A. Pease, G. Sutcliffe, First order reasoning on a large ontology, in: G. Sutcliffe, J. Urban, S. Schulz (Eds.), Proceedings of the CADE-

21 Workshop on Empirically Successful Automated Reasoning in Large Theories (ESARLT), Vol. 257 of CEUR Workshop Proceedings, CEUR-WS.org, 2007.
[63] P. Hayes, C. Menzel, A semantics for the knowledge interchange format, in: IJCAI 2001 Workshop on the IEEE Standard Upper Ontology, 2001.
[64] A. Pease, C. Benzmüller, Sigma: An integrated development environment for formal ontology, AI Communications (Special Issue on Intelligent Engineering Techniques for Knowledge Bases), to appear.
[65] P. B. Andrews, An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof, 2nd Edition, Kluwer Academic Publishers, 2002.
[66] J. R. Hindley, Basic Simple Type Theory, Cambridge University Press, 1997.
[67] P. B. Andrews, General models and extensionality, Journal of Symbolic Logic 37 (1972) 395–397.
[68] P. B. Andrews, General models, descriptions, and choice in type theory, Journal of Symbolic Logic 37 (1972) 385–394.
[69] L. Henkin, Completeness in the theory of types, Journal of Symbolic Logic 15 (1950) 81–91.
[70] G. Sutcliffe, The TPTP World - infrastructure for automated reasoning, in: E. M. Clarke, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers, Vol. 6355 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–12.
[71] T. Raths, J. Otten, Implementing and evaluating theorem provers for first-order modal logics, in: M. Giese (Ed.), Proceedings of FTP 2011, 2011, see also the online results at http://www.cs.uni-potsdam.de/ti/iltp/qmltp/download/QMLTP-v1.0-comparison.txt.
[72] V. Thion, S. Cerrito, M. C. Mayer, A general theorem prover for quantified modal logics, in: Proceedings of TABLEAUX 2002, Vol. 2381 of Lecture Notes in Computer Science, Springer, 2002, pp. 266–280.
[73] U. Hustadt, R. A. Schmidt, MSPASS: Modal reasoning by translation and first-order resolution, in: R. Dyckhoff (Ed.), Proceedings of TABLEAUX 2000, Vol. 1847 of Lecture Notes in Artificial Intelligence, Springer, 2000, pp. 67–71.
[74] C. Benzmüller, L. C. Paulson, Multimodal and intuitionistic logics in simple type theory, The Logic Journal of the IGPL 18 (2010) 881–892.
[75] C. Benzmüller, Combining and automating classical and non-classical logics in classical higher-order logic, Annals of Mathematics and Artificial Intelligence. In print; see http://dx.doi.org/10.1007/s10472-011-9249-7.
[76] M. Fitting, Interpolation for first order S5, Journal of Symbolic Logic 67 (2) (2002) 621–634.
[77] P. Blackburn, M. Marx, Tableaux for quantified hybrid logic, in: U. Egly, C. G. Fermüller (Eds.), Proceedings of TABLEAUX 2002, Vol. 2381 of Lecture Notes in Computer Science, Springer, 2002, pp. 38–52.
[78] T. Braüner, Natural deduction for first-order hybrid logic, Journal of Logic, Language and Information 14 (2) (2005) 173–198.
[79] C. Benzmüller, L. C. Paulson, Quantified multimodal logics in simple type theory, Logica Universalis (Special Issue on Multimodal Logics), to appear; see also arXiv e-prints 0905.2435 and 0905.4369.
[80] C. Benzmüller, L. Paulson, Exploring properties of normal multimodal logics in simple type theory with LEO-II, in: C. Benzmüller, C. E. Brown, J. Siekmann, R. Statman (Eds.), Festschrift in Honor of Peter B. Andrews on His 70th Birthday, Studies in Logic, Mathematical Logic and Foundations, College Publications, 2008.
[81] V. Akman, M. Surav, Steps toward formalizing context, AI Magazine 17 (3).
[82] L. Serafini, P. Bouquet, Comparing formal theories of context in AI, Artificial Intelligence 155 (2004) 41–67.
[83] S. Bucav, V. Buvac, I. Mason, Metamathematics of contexts, Fundamenta Informaticae 23 (3) (1995) 263–301.
[84] R. V. Guha, Context: A formalization and some applications, Ph.D. thesis, Stanford University (1991).
[85] F. Giunchiglia, L. Serafini, Multilanguage hierarchical logics or: How we can do without modal logics., Artificial Intelligence 65 (1) (1994) 29–70.
[86] C. Benzmüller, Automating access control logic in simple type theory with LEO-II, in: D. Gritzalis, J. López (Eds.), Emerging Challenges for Security, Privacy and Trust, 24th IFIP TC 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18-20, 2009. Proceedings, Vol. 297 of IFIP, Springer, 2009, pp. 387–398.
[87] C. Benzmüller, D. Gabbay, V. Genovese, D. Rispoli, Embedding and automating conditional logics in classical higher-order logic, submitted, see also ArXiv e-prints 1106.3685.
[88] C. Benzmüller, A. Pease, Progress in automating higher-order ontology reasoning, in: B. Konev, R. A. Schmidt, S. Schulz (Eds.), Proceedings of the Second International Workshop on Practical Aspects of Automated Reasoning, Edinburgh, UK, July 14, 2010.
[89] C. Benzmüller, A. Pease, Reasoning with embedded formulas and modalities in SUMO, in: A. Bundy, J. Lehmann, G. Qi, I. J. Varzinczak (Eds.), Proceedings of the ECAI-10 Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE-10), August 16-17, Lisbon, Portugal, 2010.
[90] C. Benzmüller, V. Genovese, Quantified conditional logics are fragments of HOL, in: Proceedings of The International Conference on Non-classical Modal and Predicate Logics (NCMPL), Guangzhou (Canton), China, 2011.
[91] K. Hoder, A. Voronkov, Sine qua non for large theory reasoning, in: N. Bjørner, V. Sofronie-Stokkermans (Eds.), Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings, Vol. 6803 of Lecture Notes in Computer Science, 2011, pp. 299–314.
[92] G. Sutcliffe, J. Urban, S. Schulz (Eds.), Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, no. 257 in CEUR Workshop Proceedings, 2007.
[93] A. Church, A revised formulation of the logic of sense and denotation. alternative (1), Noûs 27 (2).
[94] A. Church, Outline of a revised formulation of the logic of sense and denotation, part 2, Noûs 8.
[95] A. Church, Outline of a revised formulation of the logic of sense and denotation, part 1, Noûs 7.
[96] R. Montague, Pragmatics and intensional logic, Synthese 22 (1-2) (1970) 68–94.
[97] R. H. Thomason, Type theoretic foundations for context, part 1: Contexts as complex type-theoretic objects, in: Proceedings of CONTEXT '99, Vol. 1688 of Lecture Notes in Computer Science, Springer, London, UK, 1999, pp. 351–360.