Automated Consistency Checking of Expressive Ontologies — Beware of the Wrong Interpretation of Success!

Christoph Benzmüller* and Marco Ziener

Dahlem Center for Intelligent Systems, Freie Universität Berlin, Germany
`c.benzmueller@fu-berlin.de`, `m.ziener@fu-berlin.de`

Abstract. There have been attempts to (partially) translate expressive ontologies such as SUMO or Cyc to first-order logic and to use first-order automated theorem provers for detecting errors and inconsistencies. Claims have been made that these translation results are now ready for use within practical applications.
This paper adopts a critical attitude: The fact that a translation of an expressive ontology into a target logic (be it first-order or higher-order or modal, etc.) exists for which satisfiability can be shown does not imply that the approach is ready for application. What might still be missing is a guarantee of the faithfulness of the translation.
The issue is illustrated in connection with a (knowingly) unfaithful translation of SUMO into classical higher-order logic. The focus is on a small, provably satisfiable subset of this translated SUMO ontology. By adding some intuitively compatible and sound ABox axioms the unfaithfulness of the translation can easily be revealed with automated theorem provers. Without the ABox content, however, the problem remains undetected.
We thus argue for the extensive integration of ABox information into automated consistency checking for expressive ontologies, in particular, when logic translations are involved and when faithfulness has not or cannot be formally ensured.

1 Introduction

Expressive formal ontologies such as SUMO [14, 16] or Cyc [11] have been developed to support a wide range of practical applications. Since these might also include safety critical applications detecting errors and inconsistencies in the used ontologies is of utmost importance. For this purpose the use of automated theorem provers and model finders has been proposed.

Respective research efforts have led to multiple ontology revisions and they resulted in revised (partial) first-order views on them. Examples include Adimen-SUMO [1], TPTP-SUMO [15, 17], and first-orderized Cyc [18].

However, such first-orderized ontologies typically adopt very pragmatic attitudes towards some challenge aspects in the source ontologies. These include

* Supported by the German Research Foundation DFG (grant BE2501/9-1).

contextual statements in the tradition of McCarthy's work [12, 13], in which formulas, say `F`, are wrapped into context descriptions, e.g. `(in-context C F)`. These contexts can be of various kinds, including epistemic (e.g. an agent `A` knows or believes `F`: `(knows A F)`), doxastic (e.g. `(believes A F)`), or temporal (e.g. `F` holds during time `T`: `(holdsDuring T F)`).

Pragmatic solutions for handling these aspects are as follows: Adimen-SUMO simply excludes all affected axioms, that is, Adimen-SUMO is highly partial and it will hence answer queries incorrectly for which such contextualized axioms do play a role. TPTP-SUMO and first-orderized Cyc apply ad hoc translations to affected axioms (cf. [15, 18]). The faithfulness of these translations, however, has not been formally demonstrated. In fact, related problems in the TPTP-SUMO ontology have already been pointed out in the past [4].

Faithfulness of a translation α from source logic S to target logic T refers to the following property ($\mathit{\Gamma}$ is a set of S-formulas, $\mathit{\Phi}$ an S-formula, $\models^S$ and $\models^T$ are the semantical entailment relations of S and T):

$$\mathit{\Gamma} \models^S \mathit{\Phi} \quad \text{iff} \quad \alpha(\mathit{\Gamma}) \models^T \alpha(\mathit{\Phi})$$

Ideally the faithfulness of a logic translation should be proven first and then formally verified resp. formally ensured for the implementation. There is important and promising work in this direction, including the OntoIOp[1] initiative, LoLa [10] and the LATIN[2] project.

However, a formal assurance of the faithfulness (of the implementation) of a logic translation is often non-trivial to achieve in practice, and Adimen-SUMO or TPTP-SUMO are examples where this applies.

Another example is studied in this paper, where classical higher-order logic (HOL) [2, 8] is employed as target logic for expressive ontologies. In [4] two different mappings of SUMO into HOL (respectively, into the THF0 syntax [22] for HOL) have bee given. The first mapping, called THF0-SUMO-I in the remainder, translates contextualized formulas such as `(knows A F)` into HOL terms $(knows_{i\to o\to o} A_i F_o)$ were type i denotes the set of all individuals and type o stands for Booleans. Similar to the solution in TPTP-SUMO this mapping implicitly assumes extensional semantics for the contextualized, embedded formulas, and this is where faithfulness breaks. The second mapping in [4] appropriately identifies epistemic, doxastic and temporal context modifiers such as `knows`, `believes` and `holdsDuring` with respective modal logic connectives. That is, the second mapping is actually a mapping into quantified modal logic, which in turn is modeled in [4] as a fragment of HOL (for more details on the embedding of quantified modal logic in HOL we refer to [3]). The intention of second mapping thus has been to guarantee faithfulness. However, no formal formal proof has been given yet and a respective verification of the implementation appears very difficult to achieve.

[1] http://ontoiop.org
[2] http://trac.omdoc.org/LATIN/

The contributions of this paper are as follows:

First, we report on recent experiments with the THF0-SUMO-I translation. Extending the work in [4], this translation has meanwhile been applied to the entire SUMO ontology, including the mid-level ontology MILO and including all SUMO domain ontologies. Several errors in SUMO have been detected this way which remained undetected so far. However, the unfaithfulness of the THF0-SUMO-I translation could unfortunately not be revealed in these experiments.

As a second contribution this paper therefore investigates the following question: Given an expressive ontology such as SUMO and given some translation into a target formalism for which automated theorem provers and model finders are available (be it first-order logic, higher-order logic or modal logic), how much confidence should we have in it — in particular, with respect to a proper treatment of contextual statements — when the available reasoners for this target logic report satisfiability or, at least, when they cannot find any further inconsistencies in elaborate and comprehensive experiments? We use a simple example to illustrate that actually very little can be said based on such experiments. In fact, it might be misleading and even dangerous if test results are interpreted wrongly.

As a possible solution we propose the extensive integration of ABox information into automated consistency checking for expressive ontologies, in particular, when logic translations are involved and when faithfulness has not or cannot be formally ensured.

The remainder of the paper is structured as follows: In Section 2 we outline THF0-SUMO-I, our (knowingly incorrect) translation of SUMO into HOL. Section 3 presents some of the errors in SUMO that we have detected in our experiments with THF0-SUMO-I. These experiments were facilitated by our framework for elaborate consistency checking of expressive ontologies with HOL provers and model finders. This framework is outlined in Section 4. The main contribution of this paper is then presented in Section 5, where the focus is on a small, provably satisfiable subset of the THF0-SUMO-I ontology. By adding some intuitively compatible and sound ABox axioms, the unfaithfulness of the THF0-SUMO-I translation can now be revealed by HOL reasoners. Without such additional ABox axioms, however, the problem remains undetected. The paper is concluded in Section 6.

2 A naive translation of SUMO to THF0

The first mapping in [4], called THF0-SUMO-I in this paper, was deliberately chosen to simply map embedded formulas in SUMO axioms to HOL terms of type o (Booleans in THF0) and modal modifiers operating on these embedded formulas to functions on Booleans. For example, with this mapping the SUMO axioms

```
(=> (knows ?AGENT ?FORMULA) (believes ?AGENT ?FORMULA))    (1)
(=> (knows ?AGENT ?FORMULA) (truth ?FORMULA True))         (2)
```

are translated into the following THF0 representation:

```
% Type declarations
thf(truth,type,(truth: ($o>$o>$o))).
thf(believes,type,(believes: ($i>$o>$o))).
thf(knows,type,(knows: ($i>$o>$o))).

% Axioms
thf(ax1126,axiom,((! [FORMULA: $o,AGENT: $i]:                    (3)
 ((knows @ AGENT @ FORMULA) => (believes @ AGENT @ FORMULA))))).
thf(ax3303,axiom,((! [FORMULA: $o,AGENT: $i]:                    (4)
 ((knows @ AGENT @ FORMULA) => (truth @ FORMULA @ $true))))).
```

The THF0 formulas (3) and (4) correspond to the SUMO axioms (1) and (2), respectively. `$o` represents the type of Booleans, `$i` the type of individuals, `!` stands for universal quantification; the quantified variables together with their types are given in `[.]`-brackets. `=>` is implication and `@` is the explicit application operator as required in THF0 syntax. The types of the predicates `truth`, `believes`, and `knows` are provided in the type declaration section; `believes` and `knows` take an individual and a formula (i.e., a Boolean value `$true` or `$false`; remember that HOL is extensional) as argument and return a Boolean value, `truth` acts like a binary connective. More information on THF0 is provided in [22] and the THF0-SUMO-I translation is discussed in more detail in [4].

Extending the achievements in [4], we have applied the THF0-SUMO-I translation to produce a translation of the entire SUMO ontology, including the mid-level ontology MILO and all domains ontologies; this THF0-SUMO-I ontology can be download from:

http://www.christoph-benzmueller.de/papers/SUMOMILODOMAINS.thf

For the understanding of this paper the very details of THF0-SUMO-I are not necessarily required. Just remember that modal operators are wrongly mapped to extensional functions on type *o* (Booleans). Moreover, the instantiation (copying) of axioms for different types was required. The reason is that SUMO contains some axioms which cannot be assigned simple types otherwise. Consider, for example, the SUMO axiom `(instance instance BinaryPredicate)`, which is translated into `thf(ax,axiom,((instance_IIiioIioI @ instance_IiioI @ lBinaryPredicate_i)))`. Here we have now two differently typed copies of `instance`. Further axioms may now have to be formulated for both of these copies, etc.

3 Detecting errors in SUMO

By employing THF0-SUMO-I numerous errors in the SUMO ontology have been detected which remained undetected so far, also by the experiments conducted with Adimen-SUMO or TPTP-SUMO. Most often these errors were related to unused resp. undeclared variable names; cf. `?HORSE` and `?MOTION` in the following axiom from SUMO domain ontology *Sports.kif*:

```
(=>
  (instance ?HORSEBACK Equitation)
  (exists (?HORSE)
    (and
      (instance ?MOTION HorseRiding)
      (subProcess ?MOTION ?HORSEBACK))))
```

Typically, axioms were concerned which contain embedded formulas, such as the following one in which the last occurrence of *?RES1* is not bound. Those axioms are either not translated by the existing alternative approaches (this is the case for Adimen-SUMO) or the embedded formulas are translated simply as strings (as selectively done in TPTP-SUMO). In both cases typos and semantical errors in the embedded formulas can therefore not be detected by these approaches.

```
(=>
  (and
    (instance ?AGENT Agent)
    (potentialCustomer ?CUST ?AGENT)
    (modalAttribute
      (and
        (instance ?R Reserving)
        (destination ?R ?AGENT)) Necessity)
    (conditionalProbability
      (exists (?RES1)
        (and
          (instance ?RES1 Reservation)
          (reservingEntity ?CUST ?RES1)
          (fulfillingEntity ?AGENT ?RES1)))
      (customer ?CUST ?AGENT) ?NUM1)
    (conditionalProbability
      (not
        (exists (?RES2)
          (and
            (instance ?RES1 Reservation)
            (reservingEntity ?CUST ?RES2)
            (fulfillingEntity ?AGENT ?RES2))))
      (customer ?CUST ?AGENT) ?NUM2))
  (lessThan ?NUM2 ?NUM1))xf
```

Hence, for the automated detection of such errors in expressive ontologies with automated (higher-order or first-order) theorem provers a correct and intuitively appropriate treatment of all concepts in the mapping of the ontology is not necessarily required. Even very flawed mappings could still be useful to some extent.

"The procedure we have used for finding inconsistencies in the ontology can be sketched as follows:

1. *An automated procedure translates a large part of SUMO (and discharges the remaining axioms).*
2. *Then, the whole resulting formula is given (as input) to a theorem prover for automatically finding an inconsistency (that is, without providing a goal).*
3. *When a refutation is found, the theorem prover provides a description of the proof, from which we select the collection of axioms involved in that refutation.*
4. *With the help of theorem provers (for example, for finding minimal inconsistent subcollections of axioms), we identify the source of the inconsistency and repair it.*
5. *Once we repair the problem, the process is repeated from the beginning."*

Fig. 1. Automated error and inconsistency checking for expressive ontologies as has been applied for Adimen-SUMO; the text is copied from [1].

4 A framework for error and inconsistency detection

To enable experiments with THF0-SUMO-I (and with further mappings to HOL) we have developed and deployed a framework for the mechanized inconsistency detection in expressive ontologies with HOL theorem provers and model finders.

For this we have followed the same overall procedure as has e.g. been applied for Adimen-SUMO. This procedure, replicated from [1], is described in Fig. 1.

A main challenge in our work has been to appropriately cluster the THF0-SUMO-I ontology into smaller subsets which can still be processed by the available THF0 theorem provers and model finders.

A naive approach would be to generate all subsets of the powerset of axioms in THF0-SUMO-I and to apply the THF0 reasoners to them; thereby one could start with the unit sets and work towards larger and larger supersets.

Due to the size of the THF0-SUMO-I ontology such an exhaustive approach is practically infeasible. Therefore we have decided to cluster the THF0-SUMO-I ontology first into smaller subontologies. As in related work on ontology clustering [19, 23, 24] the idea is to identify subsets of axioms which are semantically independent to a large degree. Recent progress in ontology clustering, where partial consistency proofs may even be combined into global consistency proofs, is presented in [9].

Our work has focused on the exploitation of type information for ontology clustering. Remember that axioms are copied for differently typed instances of the same SUMO symbols in THF0-SUMO-I. Hence, we conjectured that this type information might serve as an effective differentiation criterion.

The idea is formalized by viewing the ontology as a structure $\mathbf{O}(\mathbf{T}, \mathbf{A}, \mathbf{R})$ where

– **T** is the set of all types used in the ontology (as mentioned before, the THF0-SUMO-I translation actually generates many different types and it often introduces axiom duplicates for different types);
– **A** is the set of all axioms used in the ontology;
– **R** is the set of user defined relations on a combination of the two previous sets or restricted to one of them.

Given a relation $r \in \mathbf{R}$, our framework computes the induced graph. For inconsistency detection we consider all possible subgraphs of this graph. In order to generate a valid THF0 input file an iteration over all the axioms and types of the subgraph is needed together with some removal of duplicates. By default relation r is chosen such that $r(a, a')$ holds (for $a, a' \in \mathbf{A}$) if and only if there exists $t \in \mathbf{T}$ which is used in both a and a'. Respective properties of types and axioms, on which relations r can be defined, are meta data and they are extracted first. This meta data forms an implicit adjacency list which is stored in a database.

The framework is implemented in the Python programming language. It supports the extraction of meta data from a THF0 ontology and it integrates the HOL automated theorem provers Nitpick [6], LEO-II [5] and Satallax [7]. It is capable of correctly interpreting their results by using the SZS ontology [21]. Moreover, it provides optimizations like caching and generator expressions for increasing speed and for limiting memory consumption on subset generation. Furthermore, the framework stores the ontology and the extracted meta data in a database for persistence.

The framework supports three different execution modes: (A) It can be deployed locally allowing the user to check and develop algorithms on his local machine, which he can then deploy to the cluster. (B) It is also possible to use the framework on the TORQUE [20] cluster or (C) on a custom setup by using Celery[3]. When running locally or when using Celery each subgraph is translated into a single job to be executed. Satisfiable subgraphs as well as generated finite models are dismissed in order to save storage space.

On TORQUE the framework uses templates for job generation as well as a packaging mechanism to reduce the overhead caused by distributing the computation onto different machines. Additionally, the framework employs a two step approach when running on TORQUE. If a job consisting of many THF0 problems runs into a timeout, then this timeout will be detected and the undetermined subsets will be written into the database for further examination. If the job is successful the results will be parsed and negative results (pointing to errors and inconsistencies) will be stored in the database. The database is polled on a regular basis for jobs with timeouts and the framework spawns a single instance of this job with a different prover in order to derive a result.

The error detection framework has been extensively applied over several weeks to the THF0-SUMO-I ontology. A large number of proof problems (to be precise, 3.047.128 at the time of writing) has been generated and passed to the above THF0 reasoners. Thereby, the errors as mentioned in Section 3 have been detected.

[3] http://www.celeryproject.org/

Nevertheless, the outcome of the experiments was disappointing to us: Our expectation was that the (known) unfaithfulness of the THF0-SUMO-I translation could eventually be experimentally revealed. Since this was not the case we did conduct some further experiments in order to find out why this was not the case. These additional experiments are summarized and discussed in the next section.

5 Unfaithfulness may be hard to detect

Can the unfaithfulness of a translation of an expressive ontology into a target logic be automatically detected with automated reasoners for this target logic in experiments? Or, alternatively, what does it mean if extensive experiments in this sense stabilize without revealing any further errors or inconsistencies? Can we then trust the translation? What does it imply if we can even formally prove the consistency of the translation result?

Statements as we find them, for example, on the Adimen-SUMO website (cf. http://adimen.si.ehu.es/web/AdimenSUMO) indicate that stabilizing experiments (or satisfiability results) in this sense are in fact applied as a criterion for the trustfulness of a translation:

As a result of this process [meant is the process as described in Fig. 1]*, we obtain a validated and consistent first-order version of the ontology to be used by first-order theorem provers.*

As outlined in Sections 3 and 4 , we have applied a related process to THF0-SUMO-I. Hence, had we not already been aware of the unfaithfulness of the translation, the temptation would have been huge to make a similar statement for the THF0-SUMO-I ontology.

Sure, THF0 theorem provers and model finders are still comparably weak, and one might argue that this is the reason for not detecting the unfaithfulness of the translation (note that showing the satisfiability of the entire THF0-SUMO-I ontology is currently still way beyond the computational capabilities of the higher-order model finders Nitpick, Satallax and LEO-II). However, as we will illustrate next, this is not the crucial point. Even if satisfiability of the entire THF0-SUMO-I ontology could be formally shown, this would still not imply that THF0-SUMO-I can now be safely employed in applications.

For demonstrating this we consider a SUMO toy ontology consisting of exactly the two SUMO axioms (1) and (2) from page 3; this ontology is called **E1** in the remainder. The THF0-translation of **E1** consists of the axioms (3) and (4) as given on page 4 (together with the necessary type declarations). This subontology of THF0-SUMO-I is called **E1***.

When being applied to **E1*** the THF0 reasoners LEO-II, Satallax and Nitpick report satisfiability (LEO-II and Satallax do this within a few milliseconds on standard PCs, and Nitpick needs about 4 seconds).

Next, we add some ABox information to our toy ontology. Thus, we extend **E1** and **E1*** into **E2** and **E2***, respectively. The added facts express that it is not the truth that Bruce is the father of Ben and of Bill (aboxAx1), that Peter

```
% Type Declarations
thf(bill,type,( bill: $i )). thf(ben,type,( ben: $i )).
thf(bruce,type,( bruce: $i )). thf(peter,type,( peter: $i )).
thf(father,type,( father: $i > $i > $o )).
thf(truth,type,( truth: $o > $o > $o )).
thf(believes,type,( believes: $i > $o > $o )).
thf(knows,type,( knows: $i > $o > $o )).

% Axioms
thf(ax1126,axiom,(
    ! [FORMULA: $o,AGENT: $i] :
      ( ( knows @ AGENT @ FORMULA ) => ( believes @ AGENT @ FORMULA ) ) )).

thf(ax3303,axiom,(
    ! [FORMULA: $o,AGENT: $i] :
      ( ( knows @ AGENT @ FORMULA ) => ( truth @ FORMULA @ $true ) ) )).

thf(aboxAx0,axiom,
    ( ( ben != bill ) & ( bruce != ben ) & ( bruce != bill )
    & ( peter != ben ) & ( peter != bill ) & ( peter != bruce ) )).

thf(aboxAx1,axiom,(
    ~ ( truth
      @ ( ( father @ bruce @ ben ) & ( father @ bruce @ bill ) )
      @ $true ) )).

thf(aboxAx2,axiom,
    ( knows @ peter @ ( father @ bruce @ ben ) )).

thf(aboxAx3,axiom,
    ( believes @ peter @ ( father @ bruce @ bill ) )).
```

Fig. 2. The satisfiable toy ontology **E2***. Adding `~(believes peter ~(father bruce bill))` results in an unsatisfiable set of THF0 axioms. This is clearly counter-intuitive.

knows that Bruce is the father of Ben (aboxAx2), and that Peter believes that Bruce is the father of Bill (aboxAx0); in SUMO notation these axioms read as:

`~(truth ((father bruce ben) & (father bruce bill)) True)` (aboxAx1)

`(knows peter (father bruce ben))` (aboxAx2)

`(believes peter (father bruce bill))` (aboxAx3)

Axiom (aboxAx0) additionally states that the constant symbols `peter`, `bruce`, `ben` and `bill` denote mutually different individuals. The detailed content of **E2*** is presented in Fig. 2.

Toy ontology **E2*** is still satisfiable, which is quickly confirmed by LEO-II, Satallax and Nitpick. This is also what we intuitively expect: Peter's believes may well differ from the true facts about the world.

Next, we slightly reformulate the content of axiom (aboxAx3) and state that Peter does not believe that Bruce isn't the father of Bill:

```
~(believes peter ~(father bruce bill))
```
(aboxAx4)

The translation of this SUMO axiom to THF0 is (aboxAx4*):

```
thf(aboxAx4,axiom,
    ( ~ ( believes @ peter @ ( ~ ( father @ bruce @ bill ) ) ) )).
```

According to our intuition and according to the intended semantics of SUMO, axiom (aboxAx4*) should be consistent with **E2***. However, this is not the case: now LEO-II, Satallax and Nitpick do report unsatisfiability. They also report unsatisfiability when (aboxAx3) is being removed from the example. Moreover, LEO-II and Satallax show that the query

```
believes peter ~(father bruce bill))
```
(query)

is implied by **E2***.

The latter results do obviously contradict our intuition on knowledge and belief, and they do also contradict the semantics of these modalities as intended in SUMO. The problem clearly is the unfaithfulness (more precisely, the wrongly assumed fully extensional semantics) of the THF0-SUMO-I translation. However, as our example nicely illustrates: it cannot be expected that the unfaithfulness of such a logic translation can be revealed without adding further ABox information and/or queries.

6 Conclusion

The unfaithfulness of a translation of an expressive ontology into a target logic might not be detectable by automated reasoners for this target logic in experiments. In our example case that was caused by an extensional treatment of intensional notions, and this mismatch could not be revealed by solely studying the translated ontology. However, it has been illustrated that the addition of further ABox information and of user queries may eventually help.

Instead of conducting theorem prover supported (in-)consistency checking on the translations of the terminological content of expressive ontologies only, we instead argue for the additional exploitation of annotated test corpora within experiments. These test corpora should ideally contain substantial, well selected ABox information and related user queries. As we have documented, the latter approach can eventually reveal inconsistencies and unfaithful translations which will remain undetected otherwise.

Allowedly, the faithfulness of a translation should ideally be formally proved before the translation is implemented and deployed, and in this respect the work as pursuit in the OntoIOp, LoLa and LATIN projects is promising and important. However, such an approach may not be always easily feasible. In fact, neither for Adimen-SUMO nor for TPTP-SUMO such proofs have been provided. Moreover, even if a respective pen and paper proof is given, typically little can be implied from it for the faithfulness of the actual implementation.

Future work includes studying the following question: How can the systematic creation of respective ABox- and user query-enriched test corpora for expressive ontologies be achieved in practice? Can such a process eventually be (partially) automated?

Acknowledgments: We thank Jasmin Blanchette, Chad Brown, Adam Pease and Geoff Sutcliffe for the provers and the tools which we have employed in our work. Moreover, we thank the anonymous reviewers of this paper for their valuable feedback.

References

1. Javier Álvez, Paqui Lucio, and German Rigau. Adimen-sumo: Reengineering an ontology for first-order reasoning. *Int. J. Semantic Web Inf. Syst.*, 8(4):80–116, 2012.
2. Peter Andrews. Churchś type theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2009 edition, 2009.
3. Christoph Benzmüller and Lawrence Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.
4. Christoph Benzmüller and Adam Pease. Higher-order aspects and context in SUMO. *Journal of Web Semantics (Special Issue on Reasoning with context in the Semantic Web)*, 12-13:104–117, 2012.
5. Christoph Benzmüller, Frank Theiss, Lawrence Paulson, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic (system description). In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *LNCS*, pages 162–170. Springer, 2008.
6. Jasmin Christian Blanchette and Tobias Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6172 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2010.
7. Chad E. Brown. Satallax: An automatic higher-order prover. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 111–117. Springer, 2012.
8. Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
9. Oliver Kutz and Till Mossakowski. A modular consistency proof for dolce. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011.* AAAI Press, 2011.

10. Christoph Lange, Till Mossakowski, and Oliver Kutz. Lola: A modular ontology of logics, languages, and translations. In *Proceedings of the 6th International Workshop on Modular Ontologies, Graz, Austria, July 24, 2012*, volume 875 of *CEUR Workshop Proceedings*, 2012.
11. Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An Introduction to the Syntax and Content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.
12. John McCarthy. Generality in artificial intelligence. *Communications of the ACM*, 30(12):1030–1035, 1987.
13. John McCarthy. Notes on formalizing context. In *Proceedings of IJCAI'93*, pages 555–562, 1993.
14. Ian Niles and Adam Pease. Towards A Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, pages 2–9, 2001.
15. A. Pease and G. Sutcliffe. First Order Reasoning on a Large Ontology. In *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories*, number 257 in CEUR Workshop Proceedings, pages 59–69, 2007.
16. Adam Pease. *Ontology — A Practical Guide.* Articulate Software Press, 2011.
17. Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. Large theory reasoning with SUMO at CASC. *AI Communications*, 23(2-3):137–144, 2010.
18. Deepak Ramachandran, Pace Reagan, and Keith Goolsbey. First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology. In *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.
19. Domenico Rosaci. An ontology-based two-level clustering for supporting e-commerce agents' activities. In *Proceedings of the 6th international conference on E-Commerce and Web Technologies*, EC-Web'05, pages 31–40, Berlin, Heidelberg, 2005. Springer-Verlag.
20. Garrick Staples. TORQUE - TORQUE resource manager. In *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing, November 11-17, 2006, Tampa, FL, USA*, page 8. ACM Press, 2006.
21. Geoff Sutcliffe. The SZS Ontologies for Automated Reasoning Software. In *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, pages 38–49, 2008.
22. 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.
23. Donato Malerba Valentina A.M. Tamma, Pepijn R.S. Visser and Dean M. Jones. Computer assisted ontology clustering for knowledge sharing. In *Proceedings of ECML2000/ML net Workshop on Machine Learning in the New Information Age*, page 7583, 2000.
24. Pepijn Visser and Valentina Tamma. An experience with ontology clustering for information integration. In *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration*, volume 23 of *CEUR Workshop Proceedings*, 1999.