Evaluating Automated Theorem Provers Using Adimen-SUMO*

Javier Álvez[1], Paqui Lucio[1], and German Rigau[1]

University of the Basque Country UPV/EHU
javier.alvez@ehu.eus
paqui.lucio@ehu.eus
german.rigau@ehu.eus

Abstract

We report on the results of evaluating the performance automated theorem provers using Adimen-SUMO. The evaluation follows the adaptation of the methodology based on competency questions [6] to the framework of first-order logic, which is presented in [3], and is applied to Adimen-SUMO [1]. The set of competency questions used for this evaluation has been semi-automatically generated from a small set of semantic patterns and the mapping of WordNet to SUMO, also introduced in [3]. Our experimental results demonstrate that improved versions of the proposed set of competency questions could be really valuable for the development of automated theorem provers.

1 Introduction

State-of-the-art automated theorem provers (ATPs) for first-order logic (FOL) are highly sophisticated systems that have been proved to provide advanced reasoning support to expressive ontologies. Since 1993, many researchers have used the *Thousands of Problems for Theorem Provers* (TPTP) problem library as an appropriate and convenient basis for ATP system evaluation [15], which is the de facto standard set of test problems for classical first-order ATP systems. Every year, the performance of ATP systems is evaluated in the *CADE ATP System Competition* (CASC) [12, 16] in the context of a set of problems chosen from the TPTP problem library and a specified time limit for each individual problem.

The CASC competition is divided into several divisions, among others the *LTB division*: First-order (FO) form theorems (axioms with a provable conjecture) from Large Theories, presented in Batches. The eligible problems for the LTB division belong to the *Commonsense Reasoning* (CSR) domain of the TPTP problem library. Some problems of the CSR domain are based on the *Suggested Upper Merged Ontology* (SUMO) [8]. Since SUMO is not written in a FOL language, the problems are specified by following the translation of SUMO into FOL proposed in [11], but have been not longer maintained since TPTP problem library version v5.3.0 (current TPTP version is v6.3.0) and not currently used in the CASC competition.

In [3], the authors adapt the methodology proposed in [6] for the evaluation of ontologies —which is based on competency questions (CQs)— to using ATP systems. Additionally, the authors also propose a new set of CQs that is semi-automatically obtained from WordNet [4] and its mapping to SUMO [9], which is successfully used for evaluating the competency of ontologies derived from SUMO [3, 2]. More specifically, the competency of Adimen-SUMO is evaluated in [2], which is the result of a reengineering process that enables the transformation of the top levels of SUMO into FOL formulas [1].

*This work has been partially funded by the Spanish Projects SKaTer (TIN2012-38584-C06-02) and COMMAS (TIN2013-46181-C2-2-R), the Basque Project LoRea (GIU15/30) and grant BAILab (UFI11/45).

In this paper, we report on the results of applying the method and the set of CQs described in [3] (around 7,500 goals) for evaluating the performance of several ATP systems. Concretely, we have evaluated several versions of Vampire [13], E [14] and vanHelsing [7]. Our experimentation enables to measure the performance of the above ATP systems when solving different kind of problems. Improved versions of the proposed set of CQs could be considered to be included in the CSR domain of the TPTP problem library.

In order to make the paper self-contained, we introduce the ontology Adimen-SUMO, the evaluation method and the benchmark in the next three sections. Then, we introduce the ATP systems that are evaluated in Section 6. In the last section, we provide some concluding remarks.

2 Adimen-SUMO

SUMO[1] [8] has its origins in the nineties, when a group of engineers from the IEEE Standard Upper Ontology Working Group pushed for a formal ontology standard. Their goal was to develop a standard upper ontology to promote data interoperability, information search and retrieval, automated inference and natural language processing.

	Unit clauses	General clauses
Meta-knowledge	0	8
Top level of SUMO	2,034	650
Mid level of SUMO	2,601	971
FO transformation	0	1,152
Total	4,635	2,781

Table 1: Some figures about Adimen-SUMO

SUMO is expressed in SUO-KIF (Standard Upper Ontology Knowledge Interchange Format [10]), which is a dialect of KIF (Knowledge Interchange Format [5]). Both KIF and SUO-KIF can be used to write FOL formulas, but its syntax goes beyond FOL. Consequently, SUMO cannot be directly used by FOL ATPs without a suitable transformation. Adimen-SUMO is the result of transforming around %88 of axioms from Top and Middle levels of SUMO (from now on, the *core* of SUMO) into FOL formulas [1]. The translation is based on a small set of axioms, which provide the axiomatization of the meta-predicates that are required to define the knowledge of SUMO as FOL formulas. Some of these meta-predicates are *instance*, *subclass*, *disjoint* and *partition*, whose axiomatization cannot be directly inherited from SUMO. The transformation also adds new axioms for a suitable characterization of SUMO types, variable-arity relations and $holds_k$ predicates, which simulate the use of variable-predicates in FOL formulas. It is worth to note that we have detected hundreds of defects of SUMO with the help of ATPs, which have been corrected in Adimen-SUMO. In Table 1, we provide some figures about the content of Adimen-SUMO. More specifically, the number of atomic formulas (unit clauses) and non-atomic formulas (general clauses) that is used in each part of Adimen-SUMO. Roughly speaking, atomic formulas (or simply atoms) make the explicit knowledge of the ontology, whereas non-atomic formulas (which contain connectives and/or quantifiers) define the implicit knowledge. It is worth to note that the axiomatization of meta-predicates (meta-knowledge) and the axioms required for the transformation into FOL formulas (FO transformation) do not

[1] http://www.ontologyportal.org

include any atomic formula. To sum up, we provide 4,635 unit clauses and 2,781 general clauses plus a conjecture as source to the ATP for each CQ.

3 Evaluation Method

In [6], the authors propose to evaluate the expressiveness of an ontology by proving completeness theorems w.r.t. a set of CQs. The proof of completeness theorems requires to check whether a given CQ is entailed by the ontology or not. In [3], the authors adapt the above method to use FOL ATPs as a tool for automatically checking whether a CQ is entailed or not. In particular, this adaptation is based on the use of FOL ATPs that work by refutation within a given execution-time limit. Following this proposal, the set of CQs is partitioned into two classes: *truth-tests* and *falsity-tests*, depending on whether we expect the conjecture to be entailed by the ontology or not. If the ATP is able to find a prove for a given conjecture, then we know for sure that the corresponding CQ is entailed by the ontology. However, if the ATP cannot find a proof, we do not know if the conjecture is not entailed by the ontology or although the conjecture is entailed, the ATP has not been able to find the proof within the provided execution-time limit. Consequently, and according to the proposal in [3], a truth-test is classified as (i) *passing* if the ATP finds a proof, since the conjecture is expected to be entailed, whereas a falsity-tests is classified as (ii) *non-passing* if a proof is found, because the conjecture is expected not to be entailed. Otherwise, if no proof is found, both truth-tests and falsity tests are classified as (iii) *unknown*. This classification provides an effective method to evaluate an ontology w.r.t. a given set of CQs. It is worth to remark that the evaluation results strongly depend on the execution-time limit of the ATP, since a CQ classified as unknown within a given execution-time limit may be classified as passing/non-passing within a longer execution-time limit.

4 Benchmark

As described in [3], the proposed benchmark has been built upon the mapping from WordNet to SUMO [9]. This mapping connects each synset of WordNet into a term of SUMO using three relations: *equivalence*, *subsumption* and *instance*. These relations are denoted by concatenating the symbols '=' (*equivalence*), '+' (*subsumption*) and '@' (*instance*) to the corresponding SUMO concept. For example, $piloting^2_n$, $education^4_n$ and $zero^1_a$ are connected to *Pilot*=, *EducationalProcess*+ and *Integer*@. Additionally, the complementary of the relations *equivalence* and *subsumption* are also used.

The mapping from WordNet to SUMO uses terms from the core of SUMO, but also from the domain ontologies. However, Adimen-SUMO only uses axioms from the core of SUMO. Thus, the mapping from WordNet to Adimen-SUMO is obtained as follows: for each WordNet synset not mapped to a term covered by Adimen-SUMO, the structural relations of SUMO (*instance*, *subclass*, *subrelation* and *subAttribute*) can be used to inherit the term of the core of SUMO to which the synset is connected. Note that it is sometimes required to modify the mapping relation. For example, the synset $frying^1_n$ is connected to the SUMO class *Frying*=, which belongs to the domain ontology *Food*. In the same domain ontology, *Frying* is defined to be subclass of *Cooking*, which is defined in the top level of SUMO. Consequently, *Frying* is not defined in the core of SUMO, but *Cooking* is. Thus, the synset $frying^1_n$ can be connected to *Cooking* in the resulting mapping. However, instead of *equivalence*, $frying^1_n$ is connected to *Cooking* by the *subsumption* mapping relation: that is, *Cooking*+.

Using the mapping from WordNet to Adimen-SUMO, the proposed set of CQs has been obtained on the basis of the information in the morphosemantic database,[2] which contains semantic relations between morphologically related nouns and verbs. From the 14 semantic relations defined in the database, 4 relations have been selected: *agent*, *result*, *instrument* and *event*. The first three ones relate a process (verb) which its corresponding agent / result / instrument (noun), from which 1,355 CQs (truth-tests) have been obtained by simply stating the same property in terms of Adimen-SUMO. For example, WordNet establishes that the *result* of $compose_v^2$ is a $composition_n^4$, which are respectively mapped to *ComposingMusic*+ and *MusicalComposition*=:

```
(exists (?X ?Y)                                                    (1)
    (and
        (instance ?X ComposingMusic)
        (result ?X ?Y)
        (instance ?Y MusicalComposition)))
```

The last relation *event* connects nouns and verbs referring to the same process. Being the same process, one can assume that both the noun and the verb should be mapped to the same class of SUMO. Thus, if the noun and the verb are mapped to different SUMO class constants, a plausible hypothesis is that the mapping is wrong. Following this criterion, 3 different conceptual patterns of questions are proposed, depending on the used mapping relations, with the purpose of detecting wrong-mappings. If both synsets are connected to two different SUMO class constants using the *equivalence* mapping relation (Event #1), it should be possible to prove that the two class constants denote different classes. For example, $kill_v^{10}$ and $killing_n^2$ are respectively connected to the SUMO classes *Death*= and *Killing*=, hence the following CQ is obtained:

```
(not                                                               (2)
    (equal Death Killing))
```

Using this first pattern, 24 CQs (truth-tests) are obtained. The second pattern of questions focuses on the case where the synsets are connected to different SUMO class constants and using different mapping relations (Event #2). That is, one synset is connected using the *equivalence* mapping relation, whereas the other synset is connected using *subsumption*. Being the mapping information less precise than in the first case, it does not suffice to prove that the classes are different. In this case, the pattern states that the class connected using *equivalence* cannot be subclass of the class connected using *subsumption*. For example, *event* relates $repair_n^1$, which is connected to *Repairing*=, and $repair_v^1$, which is connected to *Pretending*+. Therefore, the following CQ is obtained by assuming that *Repairing* cannot be subclass of *Pretending*:

```
(not                                                               (3)
    (subclass Repairing Pretending))
```

From the second pattern of questions, 350 CQs (truth-tests) are obtained. In fact, this second pattern can be seen as a particular case of the third one, where both synsets are connected using the *subsumption* mapping relation (Event #3). In this case, the pattern states that none of the connected SUMO classes can be subclass of the other one. For example, *event* relates the synsets $measure_v^4$ and $appraisal_n^1$, which are respectively connected to *Judging*+ and

[2] Available at http://wordnetcode.princeton.edu/standoff-files/morphosemantic-links.xls.

Comparing+. Consequently, the following CQ is obtained:

```
(not                                                    (4)
    (or
        (subclass Judging Comparing)
        (subclass Comparing Judging)))
```

Using this third pattern, we obtain 2,011 CQs (truth-tests) are obtained.

In total, 3,740 truth-tests are obtained using the above 4 semantic patterns. Additionally, by simply negating the proposed CQs, it is possible to obtain 3,740 falsity-tests. Hence, the proposed benchmark consists of 7,480 CQs.

5 Automated Theorem Provers

Roughly speaking, we have selected 3 different ATP systems for our evaluation, which are Vampire [13], E [14] and vanHelsing [7]. Next, we describe those systems and justify our selection.

On one hand, we have selected two of the most successful ATP systems in the CASC competition. The first one is Vampire,[3] a ATP system for first-order classical logic which has been the winner of the FOF — First-Order Form non-propositional theorems (axioms with a provable conjecture)— and LTB divisions in CASC during several years. Vampire implements the calculi of ordered binary resolution and superposition for handling equality, and it also implements the Inst-gen calculus. Vampire uses various standard redundancy criteria and implements several simplification techniques for pruning the search space, such as subsumption, tautology deletion, subsumption resolution and rewriting by ordered unit equalities. The reduction ordering is the Knuth-Bendix Ordering. In particular, we evaluate three different versions of Vampire: v2.6, v3.0 and v4.0. Vampire v2.6 is the CASC-J6 (2012), CASC-24 (2013) and CASC-J7 (2014) FOF division winner, and also the CASC-J6 (2012) LTB division winner. Vampire v3.0 won the 2nd place in the CASC-24 (2013) FOF division, but performing faster than the winner (Vampire v2.6), and has been used for the experimentation reported in [3, 2]. Vampire v4.0 is the CASC-25 (2015) FOF, FNT (First-order form non-propositional Non-Theorems), EPR (Effectively PRopositional clause normal form theorems and non-theorems) and LTB divisions winner. The second system that we evaluate is E, a theorem prover for full first-order logic with equality which consists of an (optional) clausifier for pre-processing full first-order formulae into clausal form, and a saturation algorithm implementing an instance of the superposition calculus with negative literal selection and a number of redundancy elimination techniques. Among other awards, E has been one of the top three ATP systems in the FOF division of CASC since 2012. E has also been used as a subcomponent by some other competitors in CASC. For its evaluation, we use E v1.9, which is the last version available at `http://www.eprover.org`.

On the other hand, we have selected an ATP system that has never participated in CASC: vanHelsing. This last ATP system is a fully automated proof checker for a subset of first-order problems tailored to a class of problems that arise in compiler verification. vanHelsing accepts input problems formulated in a subset of the TPTP language and is optimized to efficiently solve expression equivalence problems formulated in first-order logic. It is worth to mention that vanHelsing accepts all the axioms of Adimen-SUMO, and also all the conjectures proposed in our benchmark. An interesting feature of vanHelsing is that it provides graphical debugging help which makes the visualization of problems and localization of failed proofs much

[3] `http://www.vprover.org`

easier. This feature could be very useful for the debugging and maintance of Adimen-SUMO, its mapping to WordNet and its linking with other information resources. In our evaluation, we have used vanHelsing v1.0, which can be downloaded from https://github.com/VanHElsing/VanHElsing.

6 Evaluation results

In this section, we report the results of the evaluation of Vampire, E and vanHelsing using Adimen-SUMO and the framework introduced in the previous sections. For this evaluation, we have used an Intel® Xeon® CPU E5-2640 v3 @ 2.60GHz with 2GB of RAM memory per processor and an execution-time limit of 600 seconds. The ontology Adimen-SUMO, the set of CQs and the execution reports are freely available at http://adimen.si.ehu.es/web/AdimenSUMO.

Problem category	ATP systems									
	Vampire v2.6		Vampire v3.0		Vampire v4.0		E v1.9		vanHelsing v1.0	
Truth-tests (3,740)	399	10.67%	405	10.83%	189	5.05%	256	6.84%	70	1.87%
Relation (1,355)	202	14.91%	187	13.80%	86	6.35%	47	3.47%	7	0.52%
Event #1 (24)	7	29.17%	4	16.67%	2	8.33%	3	12.50%	1	4.17%
Event #2 (350)	111	31.71%	117	33.43%	32	9.14%	47	13.43%	22	6.29%
Event #3 (2,011)	79	3.93%	97	4.82%	69	3.43%	159	7.91%	40	1.99%
Falsity-tests (3,740)	553	14.79%	552	14.76%	538	14.39%	547	14.63%	548	14.65%
Relation (1,355)	17	1.25%	16	1.18%	3	0.22%	13	0.96%	12	0.89%
Event #1 (24)	0	-	0	-	0	-	0	-	0	-
Event #2 (350)	83	23.71%	83	23.71%	83	23.71%	82	23.43%	83	23.71%
Event #3 (2,011)	453	22.53%	453	22.53%	452	22.48%	452	22.48%	453	22.53%
Total (7,480)	952	12.73%	957	12.79%	727	9.72%	803	10.74%	618	8.26%

Table 2: Number/Percentage of solved problems

In Table 2, we provide the number and percentage of solved problems per category. With a small difference to Vampire v2.6, Vampire v3.0 is the winner according to the total number of solved problems, and it is also the winner in the truth-tests and Event #2 problem categories, while Vampire v2.6 is the winner in the Relation and Event #1 problem categories. E is classified in the third place and it is the winner of the Event #3 categories. Finally, Vampire v4.0 and vanHelsing occupy the last two places. It is worth to mention that there are minimal differences in the Falsity-tests problem category, which does not enable a proper comparison of the ATP systems. This problem is mainly due to the presence of many defects in the mapping from WordNet to SUMO, which are going to be corrected with the help of ATP systems. Also note that the total number of solved problems is relatively large for several reasons. Firstly, it is only possible to solve 3,740 problems among the considered 7,480 conjectures, since the second half of conjectures (Falsity-tests problem category) is obtained as negation of the first half (Truth-test problem category).[4] Consequently, both Vampire v2.6 and Vampire v3.0 solve more than a quarter of the proposed problems. Secondly, the execution-time limit (600 seconds) is relatively short regarding the size of Adimen-SUMO, which contains more than 7,000 axioms. Thirdly, ATP systems have not been optimized for the proposed set of conjectures. And, finally, the set of conjectures has not been adapted in order to obtain better results. Due to all the above reasons, we think that all the ATP systems perform successfully in our evaluation.

[4] A satisfiable formula does not entail both a conjecture and its negation by separate.

Problem category	ATP systems			
	Vampire v2.6	Vampire v3.0	Vampire v4.0	E v1.9
Truth-tests	68.20 s.	139.70 s.	106.71 s.	294.52 s.
Relation	3.97 s.	23.99 s.	57.84 s.	345.73 s.
Event #1	201.91 s.	43.22 s.	474.99 s.	281.4 s.
Event #2	67.18 s.	50.21 s.	194.23 s.	301.32 s.
Event #3	222.02 s.	474.69 s.	116.36 s.	277.62 s.
Falsity-tests	5.17 s.	62.60 s.	32.29 s.	102.42 s.
Relation	156.60 s.	178.70 s.	56.66 s.	104.84 s.
Event #1	-	-	-	-
Event #2	0.36 s.	0.53 s.	3.87 s.	4.77 s.
Event #3	0.37 s.	0.95 s.	6.23 s.	11.27 s.
Total	**31.59 s.**	62.60 s.	32.29 s.	102.42 s.

Table 3: Average run times (in seconds)

On the contrary, if we consider the average run times required for solving a problem as provided in Table 3,[5] Vampire v2.6 is the winner in the final classification, and also in the Truth-tests, Relation and Event #3 problem categories, whereas Vampire v3.0 is classified in the second position. Curiously, Vampire v3.0 performed faster that Vampire v2.6 in the CASC competition. The last two possitions are occupied by Vampire v4.0 and E. However, it is more interesting to see that the are significant differences between problem categories —also in the Falsity-tests category—, which proves that the performance of ATP systems strongly depends on the type of problems to be solved. In particular, the differences are specially large in the case of the Truth-tests category. We think that this feature is essential for any ATP system benchmark.

Problem category	ATP systems				
	Vampire v2.6	Vampire v3.0	Vampire v4.0	E v1.9	vanHelsing v1.0
Truth-tests	661 (136)	615 (139)	366 (82)	428 (83)	199 (34)
Relation	388 (100)	367 (98)	205 (66)	149 (42)	20 (9)
Event #1	76 (18)	38 (8)	22 (4)	32 (11)	13 (4)
Event #2	264 (37)	283 (49)	129 (17)	169 (32)	100 (20)
Event #3	207 (15)	155 (25)	125 (18)	267 (38)	127 (18)
Falsity-tests	410 (37)	397 (32)	340 (12)	379 (20)	369 (21)
Relation	101 (36)	82 (27)	20 (11)	149 (16)	55 (16)
Event #1	- (-)	- (-)	- (-)	- (-)	- (-)
Event #2	111 (1)	109 (1)	111 (1)	109 (1)	111 (1)
Event #3	295 (1)	300 (5)	291 (1)	304 (4)	298 (5)
Total	842 (158)	800 (157)	584 (89)	628 (89)	465 (44)

Table 4: Number of used axioms (non-atomic axioms)

In Table 4, we report the amount of different axioms that are used to refute all the conjectures. It is worth to note that we have removed all the spurious input axioms[6] from proofs

[5] vanHelsing is not included in the table since the system does not provide final run time (it only provides the time limit that is used for each strategy).

[6] We consider than an axiom reported in a proof is spurious if it is not stricly necessary to refute the conjecture.

Problem category	ATP systems				
	Vampire v2.6	Vampire v3.0	Vampire v4.0	E v1.9	vHelsing v1.0
Truth-tests	13.01 (5.26)	12.86 (5.47)	13.23 (5.56)	13.6 (6.31)	13.06 (6.20)
Relation	11.96 (5.07)	11.58 (4.99)	10.21 (4.73)	10.83 (5.06)	8.71 (4.86)
Event #1	16.43 (5.86)	14.75 (5.50)	14.00 (4.00)	14.33 (6.00)	13.00 (4.00)
Event #2	14.71 (5.90)	14.68 (6.11)	14.09 (5.41)	14.57 (6.98)	14.18 (6.45)
Event #3	13.03 (4.78)	13.06 (5.61)	16.58 (6.70)	14.11 (6.49)	13.2 (6.35)
Falsity-tests	3.11 (0.78)	3.09 (0.76)	2.88 (0.64)	3.05 (0.73)	3.02 (0.72)
Relation	11.41 (6.00)	11.06 (5.69)	9.00 (5.33)	10.69 (5.38)	10.58 (5.42)
Event #1	- (-)	- (-)	- (-)	- (-)	- (-)
Event #2	2.42 (0.52)	2.43 (0.52)	2.42 (0.52)	2.40 (0.51)	2.42 (0.52)
Event #3	2.93 (0.63)	2.93 (0.64)	2.92 (0.63)	2.94 (0.63)	2.93 (0.64)
Total	7.26 (2.66)	7.23 (2.76)	5.57 (1.92)	6.41 (2.51)	4.16 (1.34)

Table 5: Average number of different axioms (non-atomic axioms) in each proof

with the help of ATPs until obtaining an irreducible provable set axioms. By the way, we have also checked that the given conjecture is non-spurious in all the proofs obtained during our experimentation, which is a necessary requirement for any ATP system benchmark. Although Vampire v3.0 solves more problems than Vampire v2.6, Vampire v2.6 uses a larger amount of axioms. This is due to the fact that Vampire v3.0 is the ATP system that finds the smallest sets of provable axioms (see Table 5). Also note that the ratio between the number of different axioms and proofs is higher than 1 for the general problem category, and higher than 1.5 for the Truth-test category, which indicates that the proofs constructed by ATP systems are not repeatedly based on the same small set of axioms. Additionally, the results reported in Table 5 demonstrate that it is not trivial to refute the conjectures proposed in our benchmark. On the contrary, 13 different axioms are used on average for proving each problem of the Truth-test category, from which $5 \sim 6$ are non-atomic.

7 Concluding remarks

According to our experimental results, it is possible to use Adimen-SUMO and the benchmark built upon the mapping from WordNet to SUMO to evaluate ATP systems. In particular, it is worth to remark that large coverage of the our benchmark, since all the ATP systems (except for vanHelsing) occupy the first place in some of the categories according to the amount of solved problems. Additionally, there are significant variations in average run times between different categories. Consequently, it is clear that achieving a good performance in all the categories of the proposed benchmark is a challenging task for ATP systems.

In future work, we plan to enlarge our benchmark by considering more WordNet relations but also some other knowledge resources such as FrameNet of VerbNet, which can be connected with Adimen-SUMO via the mapping from WordNet to SUMO. At the same time, we are continuously evaluating and debugging Adimen-SUMO with the help of ATPs.

References

[1] J. Álvez, P. Lucio, and G. Rigau. Adimen-SUMO: Reengineering an ontology for first-order reasoning. *Int. J. Semantic Web Inf. Syst.*, 8(4):80–116, 2012.

[2] J. Álvez, P. Lucio, and G. Rigau. Evaluating the competency of first-order ontologies. In J. M. Gómez-Pérez, editor, *Proc. of the 8th Int. Conf. on Knowledge Capture (K-CAP 2015)*. ACM, 2015.

[3] J. Álvez, P. Lucio, and G. Rigau. Improving the competency of first-order ontologies. In J. M. Gómez-Pérez, editor, *Proc. of the 8th Int. Conf. on Knowledge Capture (K-CAP 2015)*. ACM, 2015.

[4] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[5] M. R. Genesereth, R. E. Fikes, D. Brobow, R. Brachman, T. Gruber, P. Hayes, R. Letsinger, V. Lifschitz, R. Macgregor, J. Mccarthy, P. Norvig, R. Patil, and L. Schubert. Knowledge Interchange Format version 3.0 reference manual. Technical Report Logic-92-1, Stanford University, Computer Science Department, Logic Group, 1992.

[6] M. Grüninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In *Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI 1995)*, 1995.

[7] R. Lezuo, I. Dragan, G. Barany, and A. Krall. vanHelsing: A fast proof checker for debuggable compiler verification. In *Proc. of the 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 167–174. IEEE, 2015.

[8] I. Niles and A. Pease. Towards a standard upper ontology. In Guarino N. et al., editor, *Proc. of the 2nd Int. Conf. on Formal Ontology in Information Systems (FOIS 2001)*, pages 2–9. ACM, 2001.

[9] I. Niles and A. Pease. Linking lexicons and ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In H. R. Arabnia, editor, *Proc. of the IEEE Int. Conf. on Inf. and Knowledge Engin. (IKE 2003)*, volume 2, pages 412–416. CSREA Press, 2003.

[10] A. Pease. Standard Upper Ontology Knowledge Interchange Format. Retrieved June 18, 2009, from http://sigmakee.cvs.sourceforge.net/sigmakee/sigma/suo-kif.pdf, 2009.

[11] A. Pease and G. Sutcliffe. First-order reasoning on a large ontology. In Sutcliffe G. et al., editor, *Proc. of the Workshop on Empirically Successful Automated Reasoning in Large Theories (CADE-21)*, CEUR Workshop Proceedings 257. CEUR-WS.org, 2007.

[12] F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.

[13] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[14] S. Schulz. E - A brainiac theorem prover. *AI Communications*, 15(2-3):111–126, 2002.

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

[16] G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.