Sigma: An Integrated Development Environment for Logical Theories

Adam Pease[1], Christoph Benzmüller[1]

Abstract. Sigma[1,2] is an open source environment for the development of logical theories. It has been under development and regular release for nearly a decade, and has been the principal environment under which the SUMO[3] has been created. We discuss its features and evolution, and explain why it is an appropriate environment for the development of expressive ontologies in first and higher order logic.

1 INTRODUCTION

There have been many environments created to support ontology development[25]. The majority, at least in recent years, have been to support creation of lightweight ontologies or taxonomies in the OWL language.

There are a limited number of language constructs in a frame-based or description-logic language. Frames have class membership and slots. Slots can have values and restrictions. The primary language construct is the taxonomy, which lends itself easily to tree-based views and editors. This is similar to object oriented language IDEs that typically have tree views for the hierarchy, and may have visual editors that allow the user to quickly create shells of code, based on the object taxonomy. Many ontology developers start by developing their products in a lightweight ontology editor that handles frame-based languages. Ontology developers who are used to that paradigm may wonder why Sigma does not offer an editing component as the primary method for developing ontologies Most modern software engineering however takes place in a text editor. Tools are an important part of the development process, and can help improve both productivity and quality. But the complexity of a modern programming language prevents modern software development from being reduced to simple forms entry and visual editors.

Modern and expressive languages for the development of formal theories, such as SUO-KIF[19] and TPTP[14] have a similar degree of expressiveness, in a broad sense, to a modern programming language. For that reason, we believe that the appropriate role for a knowledge engineering environment is in browsing, inference, analysis and other functions, rather than, at least primarily, authoring and editing.

There is promise in creating editing modes for text editors appropriate for knowledge engineering[26]. One challenge however is that the choice of a text editor, is, for a professional programmer, a very personal, and often a very strongly held preference. To the extent that knowledge engineers are also programmers, it will be difficult to create any environment so compelling that it will cause them to switch text editors. One alternative would be to capture just a portion of the "market" by working to add appropriate modes to just one text editor. Another would be to apply very significant resources, that do not appear yet to exist in the marketplace, to create modes in several powerful editors. For these reasons also, we have focused on tools other than text editing modes.

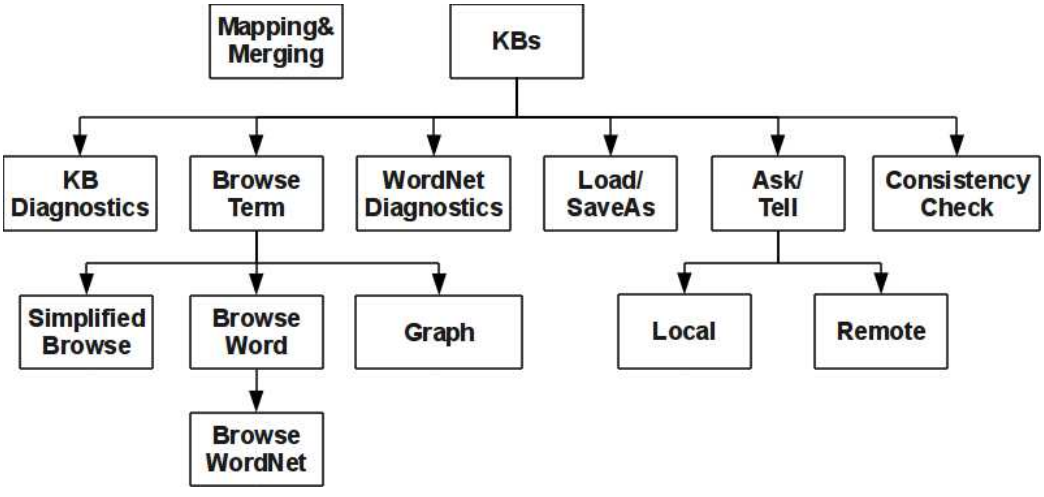

Figure 1: Major Sigma Functions

[1] Articulate Software, Angwin, CA. Email: {apease, cbenzmueller}@articulatesoftware.com.

The second author is currently supported by the German ResearchFoundation (DFG) under grant BE 2501/6-1.

Also in keeping with the modern software development model, we have utilized the Concurrent Version System (CVS) for collaborative ontology development. Developers are typically given authority over one or more ontologies, required to check in progress at least weekly so that other developers can sync up with their changes. This has also resulted in a detailed public record of the development and evolution of the Suggested Upper Merged Ontology (SUMO) [22]

While Sigma was created to support SUMO, that is by no means the only theory that it can handle. Sigma works on knowledge bases that can be composed from various files selected by the user. Those files can be coded in a small number of different formal languages, including OWL, as well as SUO-KIF. The Sigma user can easily work with very small theories or very large ones by composing only the theories that are needed for the work at hand. A typical use of Sigma would involve loading just the upper level of SUMO and whatever domain theory is needed for the user's chosen application area.

Tools within Sigma (Figure 1) can be broadly segmented into several groups, (1) browsing and display, (2) analysis and debugging, (3) inference, and (4) mapping, merging and translation. We describe each of these topics in the following sections, but first give a very brief introduction to the SUMO, which is the logical theory Sigma was initially developed to support.

2 SUMO

The Suggested Upper Merged Ontology [3,10] began as just an upper level ontology encoded in first order logic. The logic has expanded to include higher order elements. SUMO itself is now a bit of a misnomer as it refers to a combined set of theories: (1) SUMO "proper", the original upper level, consisting of roughly 1000 terms, 4000 axioms and including some 750 rules. (2) A MId-Level Ontology (MILO) of several thousand additional terms and axioms that define them, covering knowledge that is less general than those in SUMO. We should note that there is no objective standard for what should be considered upper level or not. All that can be said (simplistically) is that terms appearing lower in a taxonomy (more specific) are less general than those above. To avoid pointless argument about what constitutes an "upper level" term, we simply try to keep SUMO about 1000 terms with their associated definitions, and any time content is added, the most specific content, as measured by its having the lowest level in the subclass hierarchy, is, if necessary, moved to MILO or a domain ontology. (3) There are also a few dozen domain ontologies on various topics including theories of economy, geography, finance and computing. Together, all ontologies total roughly 20,000 terms and 70,000 axioms. We might also add a fourth group of ontologies which are theories that consist largely of ground facts, semi-automatically created from other sources and aligned with SUMO. These include YAGO[4], which is the largest of these sorts of resources aligned with SUMO.

SUMO has been mapped by hand to the WordNet lexicon[5]. Initially each term in SUMO proper was mapped and in later phases all WordNet synsets appearing above a frequency threshold in the Brown Corpus[7,8] were mapped to a roughly equivalent term in SUMO's lower level ontologies. If a rough equivalent didn't exist, one was created and defined. One caveat is that some words in English are vague enough to defy logical definition, so some such words still lack direct equivalences.

SUMO proper has a significant set of manually created language display templates that allow terms and definitions to be paraphrased in various natural languages, including non-western character sets. These include Arabic, French, English, Czech, Tagalog, German, Italian, Hindi, Romanian, Chinese (traditional and simplified characters). Automatically generated natural language paraphrases can be seen in the rightmost column of the screen display given as Figure 2.

Take for example that we have the SUO-KIF statement that **`(authors Dickens OliverTwistBook)`**. We have the following statements that have been coded to support the paraphrasing of statements with the authors relation.

```
(format EnglishLanguage authors "%1 is %n the &%author of %2")
(format it authors "%1 è l' &%autore di %2")
```

If a Sigma user has loaded this information in a knowledge base, and English is selected as the presentation, the user will see "Dickens is the author of Oliver Twist." next to the SUO-KIF statement. If Italian is selected, the paraphrase will be "Dickens è l'autore di Oliver Twist". Arguments to predicates are recursively substituted for the %1, %2 etc parameter variables, allowing much larger expressions to be constructed from more complex logical expressions.

The Global WordNet effort [6,9] links lexicons in many languages, following the same model of computational lexicon development as the original English WordNet. Wordnets have now been developed for some 40 languages. This rich set of cross-linguistic links that includes SUMO has the promise of being the basis for much work in language translation and linguistics generally. A simple idea for taking advantage of some of this work would be to expand the set of language translations for individual terms available for SUMO.

SUMO is defined in the SUO-KIF language[19], which is a derivative of the original Knowledge Interchange Format[20].

When we speak in this paper about a "formal theory", we mean a theory, such as SUMO, in which the meaning of any term is given only by the axioms in a logical language that use that term. In contrast, in an informal ontology, terms must be understood by recourse to human intuitions or understandings based on natural language term names, or natural language definitions.

3 BROWSING and DISPLAY

Sigma was originally just a display tool. Its original, and still most heavily used function is for creating hyperlinked sets of formatted axioms that all contain a particular term (Figure 1). Clicking on a term in turn gives a hyperlinked display of all the axioms that contain the new term. Next to each axiom is given the file and line(s) where the axiom was written. Also, shown is an automatically generated natural language paraphrase of each axiom. While the language generation component is relatively rudimentary, it gains significant power when tied to a rich ontology, in this case, SUMO. Much productive work remains to extend the functionality of this component to take into account the latest work in language generation. In particular, significant improvement would come from natural use of prepositions in paraphrasing statements about actions and the participants in actions.

In 2008 we added a simplified browser view that may be more appropriate for users who are transitioning from use of frame and description logic languages. It gives prominence to a tree view of the subclass hierarchy and presents binary relations in a simple tabular format, relegating rules to an area lower in the browser pane, and rendering them in the natural language paraphrase form only.

Sigma includes a tree browser display. In contrast to many ontologies developed in frame languages, SUMO has several hierarchies that can be used to organize and display the theory. These include hierarchies of physical parts, relations, attributes, processes and others. As such, the tree browser allows the user to select any transitive binary relation as the link by which the hierarchy display is created.

4 ANALYSIS and DEBUGGING

Sigma includes a number of specialized and general tools for ensuring ontology quality. The ultimate tool for quality checking on a formal ontology is theorem proving. However, there is no escape from the reality that on first- and higher-order theories, a theorem prover is not guaranteed to find all contradictions that exist. So in a practical system, there must be a combination of special purpose tests that are complete, and general purpose testing which is incomplete.

We will discuss theorem proving in the following section, so in this section we describe the various special case tests that we have found to be useful, and included in Sigma. While the number of possible tests is potentially infinite, there are a number of common problems that result from errors that are easy to make. The special case tests aim to cover these most common cases.

The SUMO-WordNet mappings also offer the opportunity to find problems exposed by differences in the two products. We believe that the two hierarchies should necessarily be isomorphic. A formal theory is a human engineered product, largely free of redundancy, and which can be edited to remove any kind of bias that is recognized by the developers. A formal theory can also contain concepts which are not lexicalized in any language. This is especially valuable at the upper levels, in which linguistic elements are so vague or ambiguous they cannot serve as a direct model for formalization. Being able to create new terms at will, when needed to formalize important notions in

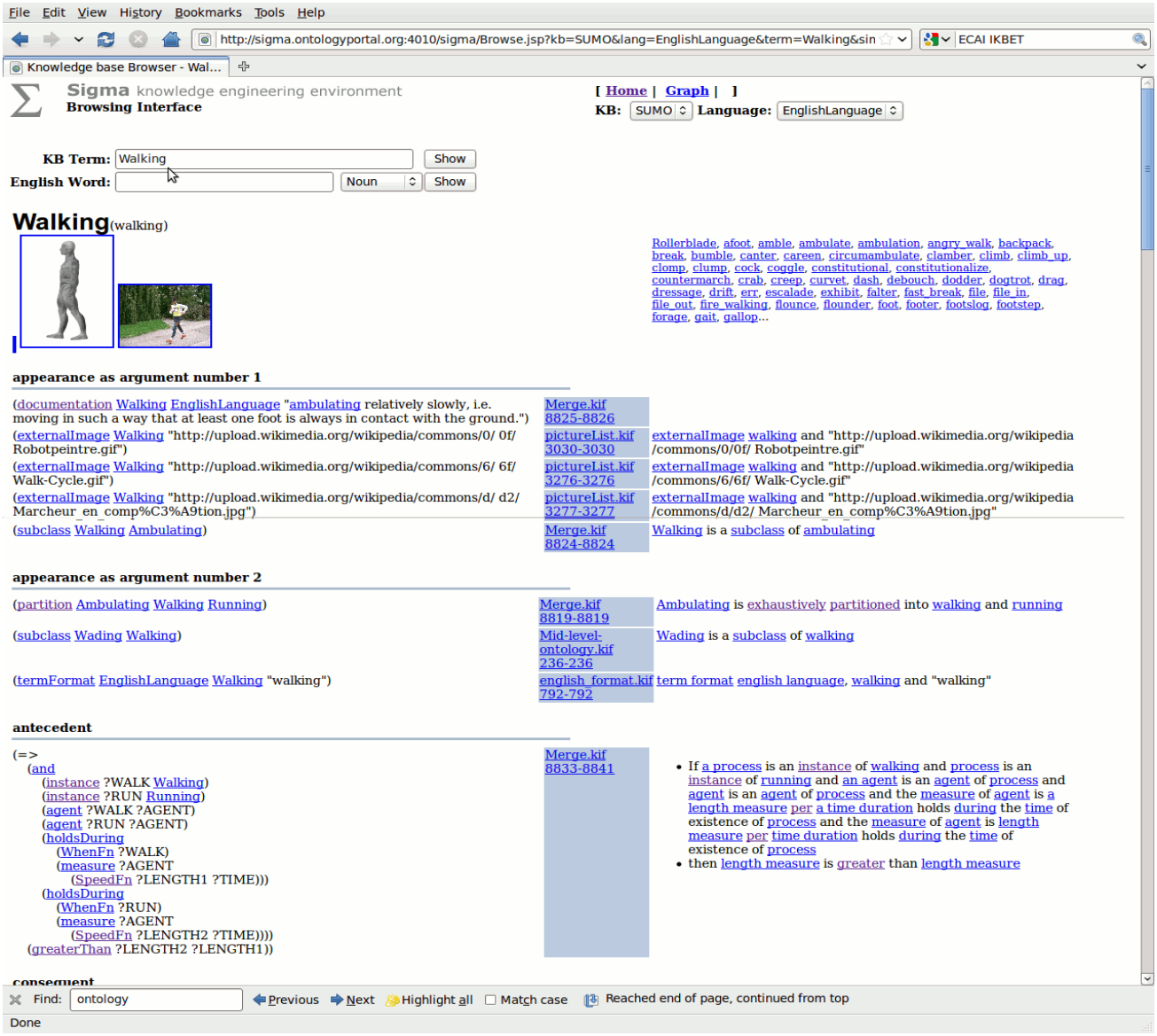

Figure 2: Sigma browsing screen

the world, is an important characteristic of a formal theory, and makes it possible to have constructs which are clear, and efficient for representation as well as inference.

There are two special case tests for errors. We test for terms without a root in the subclass hierarchy at the term Entity, which is the topmost term in SUMO. This commonly results from either omitting a subclass or instance statement when defining a new term, or by misspelling the name of the intended parent term. The second special case test is for where two terms have parents that are defined to be disjoint. In a large theory like SUMO, it can be easy to lose track of this case, especially when the ultimately conflict may be between terms that are many levels up in the subclass hierarchy.

There are also a number of tests for cases that are indicative of a problem, yet not strictly an error that would result in a logical contradiction. The first of these is for terms lacking documentation. In theories under construction, theories that are the results of importing and merging another ontology, or simply for large lists of domain instances, like city names, it may be reasonable, temporary, or expected for such terms to lack documentation. But this does often reflect an outright error, where a term name was simply misspelled in the documentation definition, or in some other axiom.

We test for cases where terms do not appear in any rules. This again is common in collections of instance-level facts, but undesirable for many terms, where it should be possible to define precisely the intended meaning of the term with a small number of formal rules, as well as statements like class membership.

Because knowledge bases are often composed from SUMO's general and domain specific component ontologies, it is desirable to limit dependencies among the files as much as possible. For that reason we include a tool to specify dependencies between pairs of files. It is typically most desirable at least to ensure that dependencies are only from one file to another, and not between both files. All domain files will of course depend at least upon SUMO proper, since they form a single integrated theory that is decomposed into separate files for convenience and efficiency of inference.

Diagnostics are provided for the SUMO-WordNet mappings. Sigma finds WordNet synsets without mapped formal terms and those for which a formal term is provided, but is not found in the current loaded knowledge base. This helps to find cases where terms have been changed or renamed and the mappings not updated. Most significant is the taxonomy comparison component. Given that we have terms A and B in SUMO and synsets X and Y in WordNet, if A is mapped to X and B to Y, Sigma checks whether if B is a subclass of A then Y is also a hyponym of X. The reverse case is also checked. It is not always the case that a hierarchy mismatch is an error. SUMO has a much richer set of relations than WordNet, as is appropriate for a formal ontology. A linguistic product must focus on linguistic relations that are directly evident in language. For example, WordNet considers a "plumber" to be a "human", whereas SUMO considers plumber to be an occupational position, and therefore an attribute that holds true about a particular human at a particular time.

5 INFERENCE

Since 2003, Sigma has used an open-source, customized version of the Vampire theorem prover called KIF-Vampire. Because SUMO has contained a limited number of higher-order constructs, and Vampire is strictly a first order prover, we have employed a number of pre-processing steps to translate SUMO into the more limited strict first order interpretation that Vampire (and other provers) can handle. These steps include (1) creating two approaches for removing variables from the predicate position. Our first approach was to add a "dummy" predicate to all clauses. This however resulted in worse performance for provers that give special indexing priority to the predicate when searching the proof space. The second approach was to instantiate every predicate variable with all possible values for predicates in the knowledge base that meet the type restrictions that may be implied by the axiom. For example,
in the following axiom, the axiom will be duplicated with the variable ?REL being instantiated with every TransitiveRelation.

```
(<=>
    (instance ?REL TransitiveRelation)
    (forall (?INST1 ?INST2 ?INST3)
        (=>
            (and
                (?REL ?INST1 ?INST2)
                (?REL ?INST2 ?INST3))
            (?REL ?INST1 ?INST3))))
```

This results in an automated expansion of the number of axioms, but does give good performance. One limitation however is that the semantics of predicate variables is thereby limited to the set of predicates existing in the knowledge base, rather than ranging over all possible predicates.

In preprocessing step (2) we turn embedded higher order formulas into uninterpreted lists of symbols. This removes most of the semantics of such statements, including the semantics of logical operators, but does at least allow for unification, thereby giving the appearance of higher order reasoning in very limited situations. In step (3) Sigma translates SUMO basic arithmetic functions into the native symbols required by KIF-Vampire. For step (4) we note that SUMO includes row variables [11], which are akin to the LISP @REST reference for variable-arity functions. We treat these as a "macro" and expand each axiom with a row variable into several axioms with one to seven variables for each occurrence of a row variable. In the few cases where axioms have two row variables, this can result in 48 new axioms. This limits the semantics of the row variables, but only to the cases that are found in practice.

Since we wish to keep Sigma as a completely open source system, we have not been able to upgrade to subsequent versions of Vampire, which are not open source, resulting in an inference component that is now somewhat out of date with respect to the state of the art. We have worked to integrate the TPTPWorld suite that has many different theorem provers, all operating under a common interface[12]. The different provers do however have different performance characteristics, and some do not provide proofs, so using this component does require a bit more expertise along with more choice. It also offers the capability to use the servers at U. Miami to run the user's inferences, which can be beneficial for those who may not have powerful computers at their location.

Integration with TPTP added a new first order language capability to Sigma for ontology reading and for export[13]. It also highlighted a limitation of Sigma until that point. Although SUMO has types defined for all relations, the logic itself is not typed. That meant that provers would not necessarily take

advantage of type restrictions in limiting their search space, and, in certain cases, could result in incorrect inferences, when inappropriate types were applied in finding solutions to queries. A theorem prover was free to use inappropriate types and then find a contradiction with SUMO's type restrictions, resulting in an inconsistent knowledge base. To solve this problem, we added a pre-processor which adds type restrictions as a new precondition to every rule. These type restrictions are deduced by collecting the most specific type restriction implied by the use of each variable as the argument to a relation in the given axiom.

Combining the automatic generation of type restrictions on axioms with the capability to generate TPTP language versions of SUMO allowed us to use SUMO-based tests in the yearly CASC competition[14,15], stretching theorem prover developers to work on high performance results in a large new category of problems in which inferences of modest difficulty must be done on a very large knowledge base, where only a small number of axioms are relevant to a given query. A key recent innovation is the SUMO Inference Engine (SInE) [16]. which selects only the subset of axioms likely to be relevant for a given query.

Another recent innovation is in translating SUMO to a typed higher order form[18] for use by true higher order theorem provers [17].The goal of this work is to better support higher order aspects in SUMO, in particular, embedded formulas and modal operators.

At the boundary of diagnostics and inference we have the general case of using theorem proving to find contradictions. Because first order proving is not guaranteed to find all problems that may exist, Sigma includes a consistency check function that leads the theorem prover to consider each axiom in a knowledge base. Each axiom is loaded one by one starting with an empty knowledge base. For each axiom, the prover is asked to computer whether the knowledge base contradicts the axiom, or is redundant with it. If the axiom doesn't create a contradiction, it is asserted to the knowledge base and the next axiom is considered. A contradiction will stop processing, since once a contradiction is found, any further results may be nonsensical. Redundancies are collected and reported once processing finishes.

Similar to the CASC competition, but on a much smaller scale, Sigma has the capability to run a series of SUMO-based tests for any theorem prover it supports, reporting success or failure and the time taken on each test.

6 MAPPING, MERGING and TRANSLATION

In addition to SUO-KIF and TPTP Sigma can also read and write OWL format [21]. Since many lightweight ontologies are currently being created in OWL, this feature opens up the use of Sigma to a large community, and provides a straightforward migration path to use of a stronger logic and more sophisticated inference. It also opens up the use of SUMO to a community that wishes to have simple and fast inference, since SUMO can be (and is) exported with a lossy translation to an OWL version. While the bulk of the SUMO axioms are not directly expressible in OWL, they can serve as informative comments (and in fact are exported as human-readable comments) that serve to better define terms for the human user than if they were simply omitted.

We should note that a general philosophy during the construction of SUMO was not to limit it to the theorem provers or techniques available at the time of knowledge engineering. If something needed to be stated to capture the semantics of a concept, we used a logic expressive enough to state it. The idea was that any statement too complicated for reasoning could at least be used as a formal concept. It's always possible to leave out complex statements in order to comply with the need for faster or decidable inference. It is not possible, obviously, to automatically create knowledge base content that does not exist, once better inference capabilities become available. This approach is paying off now that serious work is underway on practical higher order reasoning.

Sigma also includes an export of facts in Prolog form. Once Sigma generates a TPTP version of an ontology, the TPTPWorld tools also handle a translation to Prolog that supports horn clause rules. There is also a simple prototype capability for exporting SQL statements for database creation and population from Sigma.

The growing availability and coverage of lightweight taxonomies that cover domain specific knowledge, and the corresponding phenomenon of "linked data" as a community objective has encouraged the addition of an ontology mapping and merging capability to Sigma. It is based on earlier work on a stand-alone tool [23]. In mapping SUMO to simple taxonomies there is often very little information for the machine to use to determine what matches might exist. The principal problem appears to be massive numbers of false positive matches. A simple algorithm appears to do as well in practice as a more sophisticated one, since the bulk of effort is still spent by a human in selecting accurate matches. Having a simple and easy user interface appears to provide more leverage than an incrementally better matching algorithm. The Sigma matching tool has been used to create an initial alignment with the lightweight Open Biomedical Ontologies (OBO) [24] that consist mostly of taxonomic relations, with no rules and few axioms besides class membership.

7 SUMMARY and CONCLUSIONS

Sigma has served two main purposes. It is a practical tool that has supported the development of the SUMO. It is also a toolkit and testbed that is used to support experiments in ontology application and logical reasoning. Sigma has co-evolved with SUMO with each becoming more sophisticated and extensive as they progressed. The regular open source release of both products has and will continue to form a unique resource for academic and commercial researchers and practitioners engaged in ontology, natural language understanding and formal reasoning.

REFERENCES

[1] Sigma web site http://sigmakee.sourceforge.net

[2] Pease, A., (2003). The Sigma Ontology Development Environment, in Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems. Volume 71 of CEUR Workshop Proceeding series.

[3] Niles, I., & Pease, A., (2001), Toward a Standard Upper Ontology, in Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds., pp2-9.

[4] de Melo, G., Suchanek, F., and Pease, A., (2008). 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.

[5] Niles, I., and Pease, A., (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, Proceedings of the IEEE International Conference on Information and Knowledge Engineering, pp 412-416.

[6] Pease, A., and Fellbaum, C., (2010) Formal Ontology as Interlingua: The SUMO and WordNet Linking Project and GlobalWordNet, In: Huang, C. R. et al (eds.) Ontologies and Lexical Resources. Cambridge: Cambridge University Press, ISBN-13: 9780521886598.

[7] Kucera and Francis, W.N. (1967). Computational Analysis of Present-Day American English. Providence: Brown University Press.

[8] Landes S., Leacock C., and Tengi, R.I. (1998) "Building semantic concordances". In Fellbaum, C. (ed.) (1998) WordNet: An Electronic Lexical Database. Cambridge (Mass.): The MIT Press.

[9] Global WordNet web site http://www.globalwordnet.org

[10] SUMO web site http://www.ontologyportal.org

[11] Hayes, P., and Menzel, C., (2001). A Semantics for Knowledge Interchange Format, in Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology.

[12] Trac, S., Sutcliffe, G., and Pease, A., (2008) Integration of the TPTPWorld into SigmaKEE. Proceedings of IJCAR '08 Workshop on Practical Aspects of Automated Reasoning (PAAR-2008). Volume 373 of the CEUR Workshop Proceedings.

[13] Pease, A., and Sutcliffe, G., (2007) First Order Reasoning on a Large Ontology, in Proceedings of the CADE-21 workshop on Empirically Successful Automated Reasoning on Large Theories (ESARLT).

[14] Sutcliffe. G., (2007) TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 4649/2007, ISBN 978-3-540-74509-9, pp 6-22.

[15] Pease, A., Sutcliffe, G., Siegel, N., and Trac, S., (2010). Large Theory Reasoning with SUMO at CASC, AI Communications, Volume 23, Number 2-3 / 2010, Special issue on Practical Aspects of Automated Reasoning, IOS Press, ISSN 0921-7126, pp 137-144.

[16] Hoder, K. (2008) Automated Reasoning in Large Knowledge Bases, PhD thesis, Charles University, Prague, Czech Republic. See http://is.cuni.cz/eng/studium/dipl_st/index.php?doo=detail&did=49052

[17] Benzmüller, C., and Pease., A., (2010). Progress in Automating Higher Order Ontology Reasoning, Proceedings of the Second International Workshop on Practical Aspects of Automated Reasoning, Boris Konev and Renate A. Schmidt and Stephan Schulz, editors, Edinburgh, UK, July 14, 2010, CEUR Workshop Proceedings.

[18] Sutcliffe, G., and Benzmüller, C., (2010) Automated Reasoning in Higher-Order Logic using the {TPTP THF} Infrastructure, Journal of Formalized Reasoning, vol 3, no.1, pp1-27.

[19] Pease, A., (2009). Standard Upper Ontology Knowledge Interchange Format, dated 6/18/2009. Available at http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/sigma/suo-kif.pdf

[20] Genesereth, M., (1991). "Knowledge Interchange Format'', In Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, Allen, J., Fikes, R., Sandewall, E. (eds), Morgan Kaufman Publishers, pp 238-249.

[21] Sean Bechhofer, Frank van Harmelen, James A. Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein (Auth.), Mike Dean, Guus Schreiber (Ed.), OWL Web Ontology Language Reference, World Wide Web Consortium, Recommendation REC-owl-ref-20040210, February 2004.

[22] Pease, A., and Benzmüller, C., (2010) Ontology Archaeology: A Decade of Effort on the Suggested Upper Merged Ontology, in Proceeding of The ECAI-10 Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE-10), A.Bundy and J.Lehmann and G.Qi and I.J.Varzinczak editors, August 16-17, Lisbon, Portugal.

[23] Li, J., (2004) LOM: A Lexicon-based Ontology Mapping Tool, in Proceedings of the Performance Metrics for Intelligent Systems conference (PerMIS).

[24] Smith B, Ashburner M, Rosse C, Bard C, Bug W, Ceusters W, Goldberg L J, Eilbeck K, Ireland A, Mungall C J, The OBI Consortium, Leontis N, Rocca-Serra P, Ruttenberg A, Sansone S-A, Scheuermann R H, Shah N, Whetzel P L and Lewis S (2007). "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration", Nature Biotechnology 25, 1251 - 1255.

[25] Youn, S., and McLeod, D. Ontology Development Tools for Ontology-Based Knowledge Management. Encyclopedia of E-Commerce, E-Government and Mobile Commerce, Idea Group Inc., 2006.

[26] David Aspinall. Proof General: A Generic Tool for Proof Development. Tools and Algorithms for the Construction and Analysis of Systems, Proc TACAS 2000, LNCS 1785.