Knowledge Engineering for Large Ontologies with Sigma KEE 3.0

Adam Pease[1] and Stephan Schulz[2]

[1] Articulate Software, apease@articulatesoftware.com
[2] Institut für Informatik, Technische Universität München, schulz@eprover.org

Abstract. The Suggested Upper Merged Ontology (SUMO) is a large, comprehensive ontology stated in higher-order logic. It has co-evolved with a development environment called the Sigma Knowledge Engineering Environment (SigmaKEE). A large and important subset of SUMO can be expressed in first-order logic with equality. SigmaKEE has integrated different reasoning systems multiple queries to the same theory has required full re-processing of the full knowledge base.
To overcome this problem, to create a simpler system configuration that is easier for users to install and manage, and to integrate a state-of-the-art theorem prover we have now integrated Sigma with the E theorem prover. The E distribution includes a simple server version that loads and indexes the full knowledge base, and supports interactive queries via a simple interface based on text streams. No special modifications to E were necessary for the integration, so SigmaKEE can be easily upgraded to future versions.

1 Introduction

The Suggested Upper Merged Ontology (SUMO) [6, 7] is a large, comprehensive ontology stated in higher-order logic [2]. It has co-evolved with a development environment called the *Sigma Knowledge Engineering Environment* (SigmaKEE) [8].

SUMO and Sigma have been employed in applications for natural language understanding [9], database modeling [11] and sentiment analysis [10], among others.

A large and important subset of SUMO can be expressed in first-order logic with equality. This subset has been used in several instances of of the LTB division of the yearly CADE ATP System Competition (CASC)[13, 18, 14] The LTB (*Large Theory Batch*) division of CASC is concerned with reasoning in large theories, and in particular with the task of answering a series of queries over a large, relatively static background theory.

Since a large part of SUMO can be expressed in first-order logic, SigmaKEE provides first-order reasoning capabilities to support the user in interacting with the knowledge base. Earlier versions of SigmaKEE have been integrated with a customized special purpose version of Vampire [15]. However, later versions of Vampire were not compatible with the customized version and the integration

interface with SigmaKEE. As a consequence, Sigma did not have access to the latest deduction technologies.

An interim measure has been integration with the TPTPworld environment [22] that supports remote access to the entire suite of theorem provers competing in CASC, also adding facilities for generating explicit proof objects even with provers that do not natively have that capability. In recent years, an initial integration has also been done for the LEO-II higher-order logic prover [1]. However, both approaches require re-processing of the full ontology for each query, and thus result in significant overhead, resulting in noticeable delays even for simple user queries.

To overcome this problem, to use the latest first-order theorem proving technology, and to create a simpler system configuration that is easier for users to install and manage, SigmaKEE 3.0 now integrates the well-known theorem prover E [16, 17]. The E distribution includes a simple server version that loads and indexes the full knowledge base, and supports interactive queries via a simple interface based on text streams. No special modifications to E were necessary for the integration, so Sigma users will be able to upgrade to successive versions of E as they are released.

2 Architecture and User Interface

Sigma is a Java and JSP system that typically will run under the Apache Tomcat web server. It consists of a set of tools integrated at the user level by an HTML interface. These include:

- *translation* of theories to and from THF, TPTP, SUO-KIF, OWL and Prolog formats
- *mapping* theories to other theories based on similarity of terms names and definitions
- structured *browsing* of hyperlinked and sorted theory content, include tree-structured presentation of hierarchies
- *natural language paraphrases* of theories in many different languages
- structured browsing of *WordNet* [4] and Open Multilingual Wordnet [3] and their links to SUMO
- various kinds of *static analysis* tools for theories, as well as structured inference using theorem proving as a client, that attempts to find theory contradictions

The core of the system consists of data structures to manage knowledge bases, their constituent files, and the statements contained in the files. Analysis, display, natural language processing and inference components have been added incrementally around the common data structures. In particular, the facilities to handle TPTP input and output, and integration with the TPTPworld environment were added to support development for the CASC competition. Sigma is an Integrated Development Environment for ontologies in the same sense Eclipse

is an IDE for Java programming. While actual development of theories is performed in a text editor, a suite of tools assists in the process of developing text files of SUMO-based theories and a typical development process involves having both Sigma and a text editor running, and the developer frequently switching attention between the two.

By integrating E into Sigma, SUMO developers can more rapidly test new theories for consistency, as well as applying them in applications involving automated deduction. Sigma supports a hyperlinked proof output that links steps in each proof to display of the statements from which they are derived.

Users interact with Sigma via a web browser. The user interface is straightforward, consisting of a window in which queries or statements are made in SUO-KIF format, and hyperlinked proof results, which are similar to a standard textbook proof. Steps in the proof are presented along with a brief justification of how they were derived, whether directly asserted to the knowledge base, or derived via rules of inference from previous steps.

In a typical workflow, a user writes theory content in a text editor and periodically loads the file into Sigma. He will frequently use the Sigma browsing tools to inspect other portions of SUMO, for example, to find useful relations and classes to help model the knowledge of the task at hand, or simply to check relation argument orders or argument types. The user can pose queries to E to test the coverage of the new theory content. Debugging a query that doesn't yield an answer often involves breaking down a complex chain of reasoning into smaller and simpler steps, adding knowledge to complete elements of the chain and gradually building up to the full line of reasoning desired. This will involve running queries, checking proofs of lemmas, checking existing portions of SUMO to see if knowledge that might be assumed present is actually present, and in the desired form. Based on iterations of this process, the user will keep adding new statements to a theory in a text editor until the modeling or application task is complete. In typical usage, this development process is broadly quite similar to modern programming, just that the language is strictly declarative, rather than functional or procedural, and therefore the analysis and development tools themselves are different. But the cycle of development in a text editor, use of analysis, browsing and debugging tools, then further editing or development of the "program" is a familiar one.

3 SigmaKEE/E integration

SigmaKEE has been integrated with E 1.8[16, 17]. E is a powerful equational theorem prover for full first order logic with equality. It has a number of features that make it an attractive choice for this integration:

- E supports the TPTP standards for input and output. In particular, it reads specifications and writes proof objects in the TPTP-3 language [20] and uses the SZS result ontology [19] to signal success or failure of a proof attempt. Thus, the main interfaces are well-defined and results are easy to parse.

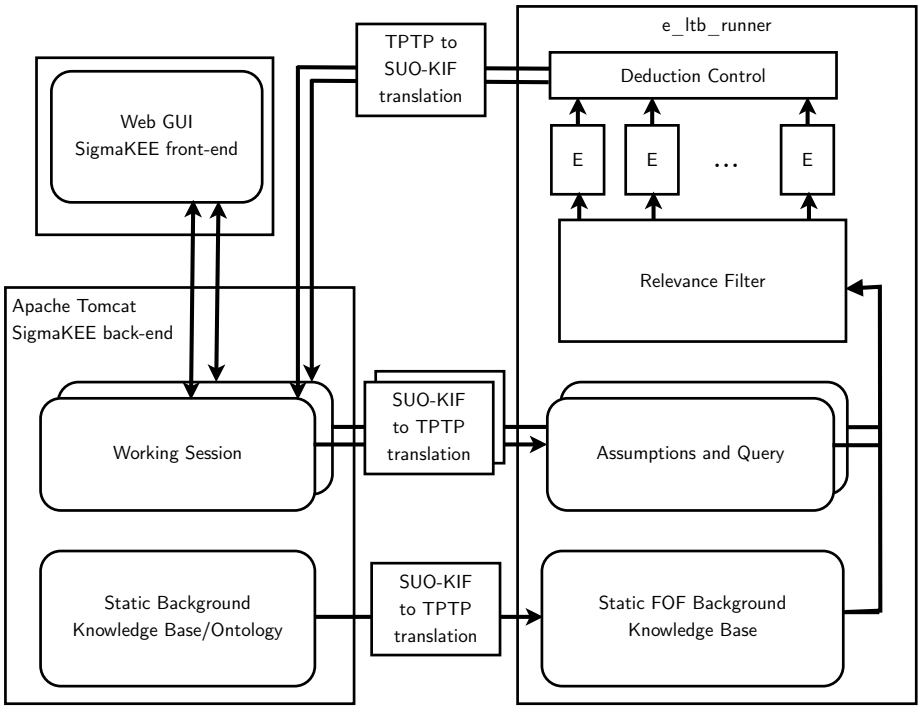

Fig. 1. Main architectural components and data flows of SigmaKEE with respect to the deduction component

- As of version 1.7, E support the proposed TPTP answer standard [21]. It can report one or multiple answers to queries, i.e. instantiations for top-level existentially quantified variables in the conjecture that make it true. This is particularly valuable for query-answering.
- With version 1.8, E can generate and print checkable proof objects with barely measurable overhead [17]. The proof objects make logical dependencies obvious, and can also help to debug the ontology, in particular by identifying inconsistencies.

In addition to support for well defined I/O standards and fast proof generation, E also provides a prototypical implementation of *deduction as a service*. The `e_ltb_runner` control program in the E distribution supports the efficient execution of multiple queries over a static background theory, both in batch mode and in interactive mode. This interactive mode forms the base of the interface between the SigmaKEE core and the deduction system.

On start-up, `e_ltb_runner` reads a specification file that describes the constant background theory, given in the form of files of clauses and formulas in TPTP-3 syntax. This initial knowledge base is indexed with a bi-directional index from formulas to function symbols and vice versa. Moreover, various statis-

tics on the distribution of function symbols in the knowledge base are computed. These pre-computed indices and values allow the efficient application of a parameterized variant of the SInE algorithm [5].

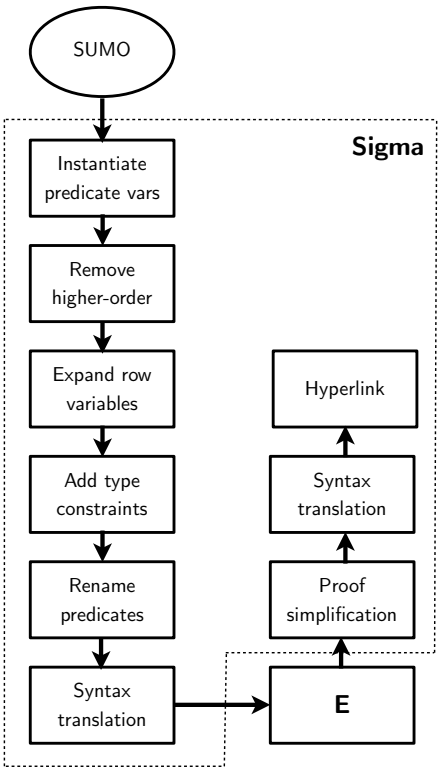

Fig. 2. Language transforms

The program then enters interactive mode and reads input from the user via `stdin`. Users can provide additional formulas, either directly or by specifying TPTP input directives to load axioms from files. Typically, a query consists of a number of additional assumptions and a conjecture (or *question*, if answer instantiations are desired). These formulas are temporarily integrated into the knowledge base and the indices are updated in a way that allows for the efficient retraction of the formulas and resetting of the indices. When the user indicates that the current specification is complete, `e_ltb_runner` runs various different relevancy filters over the extended knowledge base and extracts a number of individual proof problems, each of which contains the conjecture or query and a number of potentially useful axioms. These are handed to different instances of E in automatic mode. If one of the instances finds a proof (or a counter-saturation), all provers are stopped, and the result, along with the proof or the derivation of the saturation, is provided back on the standard output channel. If all instances of E time out or hit other resource limits, the proof attempt fails.

Once the job is processed, the additional formulas are removed from the knowledge base and the indices, and the system waits for the next user command.

Figure 1 provides an overview of the integration with SigmaKEE with E. On start-up, the SigmaKEE back-end translates the static ontology into TPTP format. It starts up `e_ltb_runner` in interactive mode, passing the translated ontology as as the background theory. The back-end connects to the deduction component via `stdin` and `stdout`.

New knowledge entered by the user is kept in a separate working session. When the user wants to query the knowledge base, the formulas of the current working session and the query are translated to TPTP syntax and provided as a job to `e_ltb_runner`. There, they are added to the background knowledge, the relevance filters are applied, and different instances of E try to find an answer to

the query. If successful, the proof is handed to the SigmaKEE back-end, where it is translated back to SUO-KIF.

Several transformations are required to convert SUO-KIF into TPTP, and to restore the content in the TPTP3 format proofs to their authored SUO-KIF versions, as shown in Figure 2. While these transforms are described in more detail in [7] and [12] a brief overview here may be helpful. First, variables that are in the predicate position in a rule are removed by instantiating every such rule with every predicate from the knowledge base that is applicable. Next, the remaining higher-order logic content that is not expressible in TPTP FOF syntax is removed. Then row-variables, which stand for multiple arguments in variable-arity relations are expanded, treating this construct as a macro. SUMO requires type constraints for the arguments to all relations. To fully implement this in a non-sorted logic such as TPTP FOF, we prefix all rules with type constraints. While in TPTP-3 implementations, any symbol identifier can be used either a function symbol of a given arity or a predicate symbol of a given arity, SUO-KIF does not share this restriction. Hence, in cases where a symbol is used in more than one role, occurrences of one type are renamed.

Lastly, the actual syntactic transform of SUO-KIF, which conforms to LISP S-Expressions is converted to the Prolog syntax of TPTP. Upon return from E, the SZS ontology tags are extracted to provide the overall status of the result. The proof is simplified to removed repeated appearances of the same statement. Answer variables are removed. The syntactic transform is now run in reverse, converting TPTP statements to SUO-KIF. Finally, predicates are returned to their originally authored names.

4 Conclusion

Sigma KEE 3.0 brings together a practical development environment for creating expressive logical theories and a leading first order theorem prover. It is a start at providing the same sort of powerful development approach for logical theories that programmers have long enjoyed for procedural and object-oriented development.

SigmaKEE and SUMO offer a development tool suite and a reusable library of content on which to build new theories. All the tools are open source, in hopes of inviting collaboration. E is available from `http://eprover.org` and as part of the Sigma distribution from `http://sigmakee.sourceforge.net`.

References

1. Benzmüller, C., Pease, A.: Progress in automating higher-order ontology reasoning. In: Konev, B., Schmidt, R., Schulz, S. (eds.) Workshop on Practical Aspects of Automated Reasoning (PAAR-2010). CEUR Workshop Proceedings, Edinburgh, UK (2010)
2. Benzmüller, C., Pease, A.: Reasoning with Embedded Formulas and Modalities in SUMO. The ECAI-10 Workshop on Automated Reasoning about Context and Ontology Evolution (August 2010)

3. Bond, F., Paik, K.: A survey of wordnets and their licenses. In: Proceedings of the 6th Global WordNet Conference (GWC 2012). Matsue (2012), 64–71
4. Fellbaum, C.: WordNet: An Electronic Lexical Database. Language, Speech, and Communication, MIT Press (1998), http://books.google.com.hk/books?id=Rehu8OOzMIMC
5. Hoder, K., Voronkov, A.: Sine Qua Non for Large Theory Reasoning. In: Bjørner, N., Stokkermans, S.V. (eds.) Proc. of the 23rd CADE, Wroclav. LNAI, vol. 6803, pp. 299–314. Springer (2011)
6. Niles, I., Pease, A.: Toward a Standard Upper Ontology. In: Welty, C., Smith, B. (eds.) Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001) (2001)
7. Pease, A.: Ontology: A Practical Guide. Articulate Software Press, Angwin, CA (2011)
8. Pease, A., Benzmller, C.: Sigma: An Integrated Development Environment for Logical Theories. AI Comm. 26, 9–97 (2013)
9. Pease, A., Li, J.: Controlled English to Logic Translation. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology. Springer (2010)
10. Pease, A., Li, J., Nomorosa, K.: WordNet and SUMO for Sentiment Analysis. In: Proceedings of the 6th International Global Wordnet Conference (GWC2012). Matsue, Japan (2012)
11. Pease, A., Rust, G.: Formal Ontology for Media Rights Transactions. In: Garcia, R. (ed.) Semantic Web Methodologies for E-Business Applications. IGI publishing (2008)
12. Pease, A., Sutcliffe, G.: First Order Reasoning on a Large Ontology. In: Proceedings of the CADE-21 workshop on Empirically Successful Automated Reasoning on Large Theories (ESARLT) (2007)
13. Pease, A., Sutcliffe, G., Siegel, N., Trac, S.: Large Theory Reasoning with SUMO at CASC. AI Comm., Special issue on Practical Aspects of Automated Reasoning 23(2-3), 137–144 (2010)
14. Pelletier, F.J., G., G.S., Suttner, C.: The development of CASC. AI Comm. 15(2-3), 79–90 (2002)
15. Riazanov, A., Voronkov, A.: The Design and Implementation of VAMPIRE. Journal of AI Communications 15(2/3), 91–110 (2002)
16. Schulz, S.: E – A Brainiac Theorem Prover. AI Comm. 15(2/3), 111–126 (2002)
17. Schulz, S.: System Description: E 1.8. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) Proc. of the 19th LPAR, Stellenbosch. LNCS, vol. 8312. Springer (2013)
18. Sutcliffe, G., Suttner, C.: The state of CASC. AI Comm. 19(1), 35–48 (2006)
19. Sutcliffe, G., Zimmer, J., Schulz, S.: TSTP Data-Exchange Formats for Automated Theorem Proving Tools. In: Sorge, V., Zhang, W. (eds.) Distributed Constraint Problem Solving And Reasoning In Multi-Agent Systems, pp. 201–215. Frontiers in Artificial Intelligence and Applications, IOS Press (2004)
20. Sutcliffe, G., Schulz, S., Claessen, K., van Gelder, A.: Using the TPTP Language for Writing Derivations and Finite Interpretations . In: Furbach, U., Shankar, N. (eds.) Proc. of the 3rd IJCAR, Seattle. LNAI, vol. 4130, pp. 67–81. Springer, 4130 (2006)
21. Sutcliffe, G., Stickel, M., Schulz, S., Urban, J.: Answer Extraction for TPTP. http://www.cs.miami.edu/~tptp/TPTP/Proposals/AnswerExtraction.html, (acccessed 2013-07-08)
22. Trac, S., Sutcliffe, G., Pease, A.: Integration of the TPTPWorld into SigmaKEE. In: Proceedings of IJCAR '08 Workshop on Practical Aspects of Automated Reasoning (PAAR-2008). CEUR Workshop Proceedings (2008)