Sigma: An Integrated Development Environment for Formal Ontology

Adam Pease, Christoph Benzmüller[1]

[1] The second author has been funded by the German Research Foundation under grant BE 2501/6-1.

Abstract. Sigma 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 open source Suggested Upper Merged Ontology (SUMO) 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

We should first discuss what we mean by a formal theory or formal ontology, as we use these terms interchangeably in this paper. For our purposes, these are mathematical entities, that are collections of statements made in a language with a formal semantics. The use of the word "ontology" may be a source of some confusion in computer science as it has been applied to so many sorts of information models that it can be almost meaningless. In particular, things which have previously been called schemas, taxonomies, semantic networks or object models have now been branded "ontologies". A key distinction for us is whether a given model has a definition in a formal language that allows each term to be interpreted without recourse to human intuition about its meaning based on the name of the term or natural language documentation about the term. Semantic web taxonomies, for example, describe their terms formally only up to isomorphism in a subclass hierarchy and typically add informal natural language comments to further characterize the intended definitions.

There have been many environments created to support ontology development [25] (in the loose sense of the phrase). The one most comparable to Sigma is the Cyc system [46], which like SUMO, includes a large ontology. Cyc contains a single inference engine, and is not open source, which limits the ability to make meaningful comparisons with other tool sets. While there are few tools comparable to Sigma in total, there are many tools which are related to the various components of Sigma and these are discussed in more detail in the section below on related research, along with references to major frame-based ontology editing tools.

The majority of ontology development tools, at least in recent years, have been created to support creation of lightweight taxonomies in the OWL language [21]. The most popular system for developing OWL, even though it pre-dates the development of that language, is Protege [45]. Developers who are familiar with languages of that sort often expect that there would exist ontology development tools that support graphically-based authoring for other languages.

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 object 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. A simple taxonomy and slot-value filling interface however would be useful for fast prototyping and will be added to Sigma in the future.

There is promise in creating editing modes for text editors appropriate for knowledge engineering, such as with the ProofGeneral environment [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.

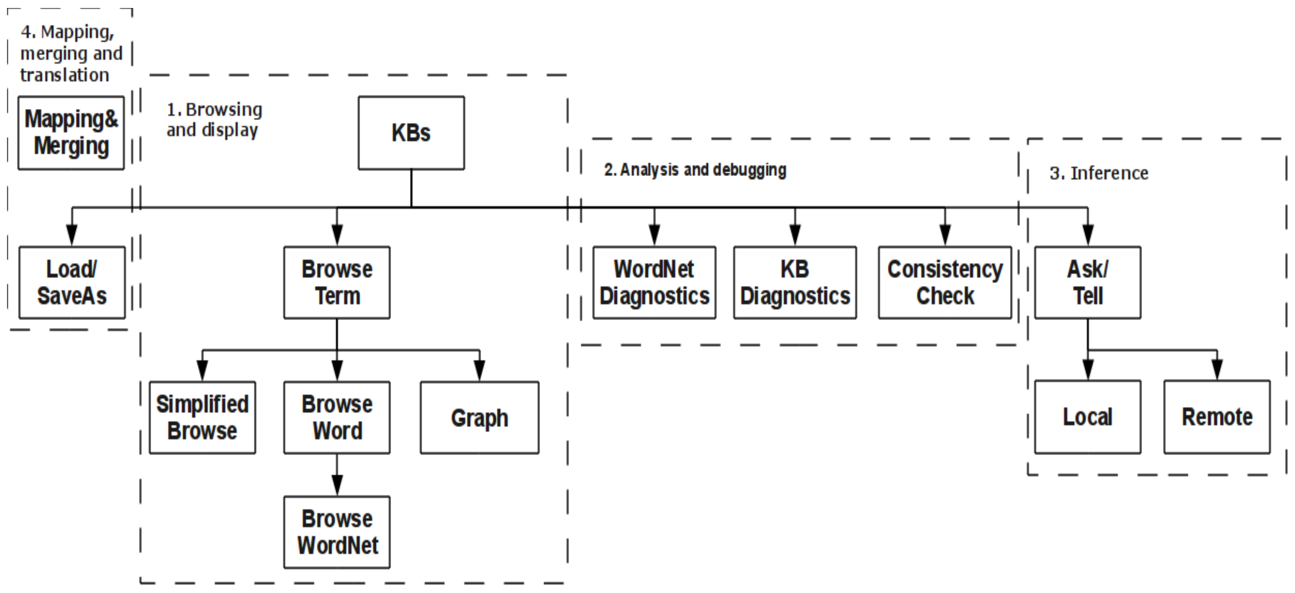


Figure 1: Major Sigma Functions

Also in keeping with a 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) [3,22].

While Sigma [1,2] was created to support SUMO, and that has been its primary use during some eight years of development, 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 TPTP and 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.

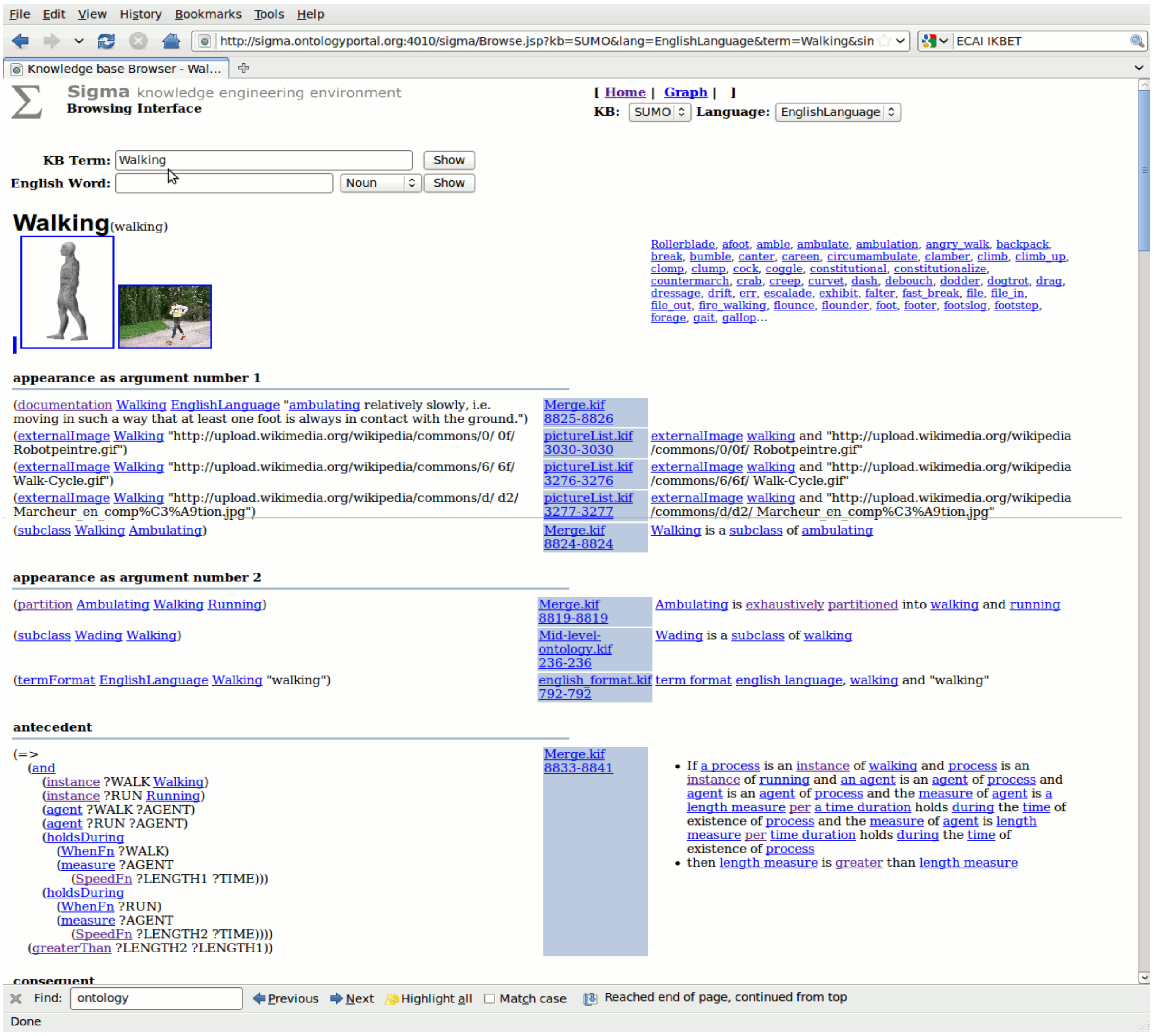
Tools within Sigma (Figure 1) can be broadly segmented into several groups, (1) browsing and display, (2) analysis and debugging, (3) inference, and (4) mapping, merging and translation. We describe each of these topics in the following sections, but first give a very brief introduction to the SUMO,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

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) the original upper level, consisting of roughly 1000 terms, 4000 axioms and including some 750 rules. In this paper, we'll refer to this portion of SUMO as SUMO "proper". (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 and has millions of facts.

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

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, and 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.


File Edit View History Bookmarks Tools Help
http://sigma.ontologyportal.org:4010/sigma/Browse.jsp?kb=SUMO&lang=EnglishLanguage&term=Walking&sin
ECAI IKBET
Knowledge base Browser - Wal...

Sigma knowledge engineering environment
Browsing Interface

[Home | Graph |]
KB: SUMO Language: EnglishLanguage

KB Term: Walking Show
English Word: Noun Show

Walking(walking)

Rollerblade, afoot, amble, ambulate, ambulation, angry_walk, backpack, break, bumble, canter, careen, circumambulate, clamber, climb, climb_up, clomp, clump, cock, coggle, constitutional, constitutionalize, countermarch, crab, creep, curvet, dash, debouch, dodder, dogtrot, drag, dressage, drift, err, escalade, exhibit, falter, fast_break, file, file_in, file_out, fire_walking, flounce, flounder, foot, footer, footslog, footstep, forage, gait, gallop...

appearance as argument number 1

(documentation Walking EnglishLanguage "ambulating relatively slowly, i.e. moving in such a way that at least one foot is always in contact with the ground.")	Merge.kif 8825-8826	
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/0/ 0f/ Robotpeintre.gif")	pictureList.kif 3030-3030	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/0/0f/ Robotpeintre.gif"
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/6/ 6f/ Walk-Cycle.gif")	pictureList.kif 3276-3276	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/6/6f/ Walk-Cycle.gif"
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/d/ d2/ Marcheur_en_comp%C3%A9tion.jpg")	pictureList.kif 3277-3277	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/d/d2/ Marcheur_en_comp%C3%A9tion.jpg"
(subclass Walking Ambulating)	Merge.kif 8824-8824	Walking is a subclass of ambulating

appearance as argument number 2

(partition Ambulating Walking Running)	Merge.kif 8819-8819	Ambulating is exhaustively partitioned into walking and running
(subclass Wading Walking)	Mid-level-ontology.kif 236-236	Wading is a subclass of walking
(termFormat EnglishLanguage Walking "walking")	english_format.kif 792-792	term format english language, walking and "walking"

antecedent

```
(=>
  (and
    (instance ?WALK Walking)
    (instance ?RUN Running)
    (agent ?WALK ?AGENT)
    (agent ?RUN ?AGENT)
    (holdsDuring
      (WhenFn ?WALK)
      (measure ?AGENT
        (SpeedFn ?LENGTH1 ?TIME)))
    (holdsDuring
      (WhenFn ?RUN)
      (measure ?AGENT
        (SpeedFn ?LENGTH2 ?TIME))))
  (greaterThan ?LENGTH2 ?LENGTH1))
```

Merge.kif 8833-8841

- If a process is an instance of walking and process is an instance of running and an agent is an agent of process and agent is an agent of process and the measure of agent is a length measure per a time duration holds during the time of existence of process and the measure of agent is length measure per time duration holds during the time of existence of process
- then length measure is greater than length measure

consequent

Find: ontology Previous Next Highlight all Match case Reached end of page, continued from top
Done

Figure 2: Sigma browsing screen

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")
```

Terms are also given language-specific strings, when appropriate

```
(termFormat EnglishLanguage OliverTwistBook
"Oliver Twist")
```

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 %n refers to the word for negation in the given language, and is inserted if the formula is negated. For example

```
(not (authors RobinCook WarAndPeace))
```

is rendered as "Robin Cook is not the author of War and Peace."

3 SUMO and WordNet

SUMO has been mapped by hand to the entire WordNet [27] lexicon [5]. WordNet consists of over 100,000 linguistic senses called "synsets" (synonym sets). For example, one WordNet synset is

mouth, speak, talk, utter, verbalise, verbalize: express in speech; "She talks a lot of nonsense"; "This depressed patient does not verbalize"

Initially, each term in SUMO proper (the 1000 term upper level of SUMO) 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, or only have meaning within the context of a sentence, so some such words still lack direct equivalences.

For example, take the word "bright" in the sense of "full of promise". In the context of "John has a bright future. He was selected for the varsity basketball team as a freshman." the word means that he is better at basketball than many of his high school classmates. In other contexts it might mean that he is more likely to recover from an orthopaedic injury than other patients of a similar demographic, or that he's likely to become president. A simple word-to-term relationship is not enough, and something more sophisticated would be needed to create a specification of meaning that is related to context. Contrast this with "walking" in the sense of ambulation. It is relatively straightforward to give some degree of formal definition the notion of "walking" as an individual term in a hierarchy of processes and differentiated from "running", "crawling", "driving" etc.

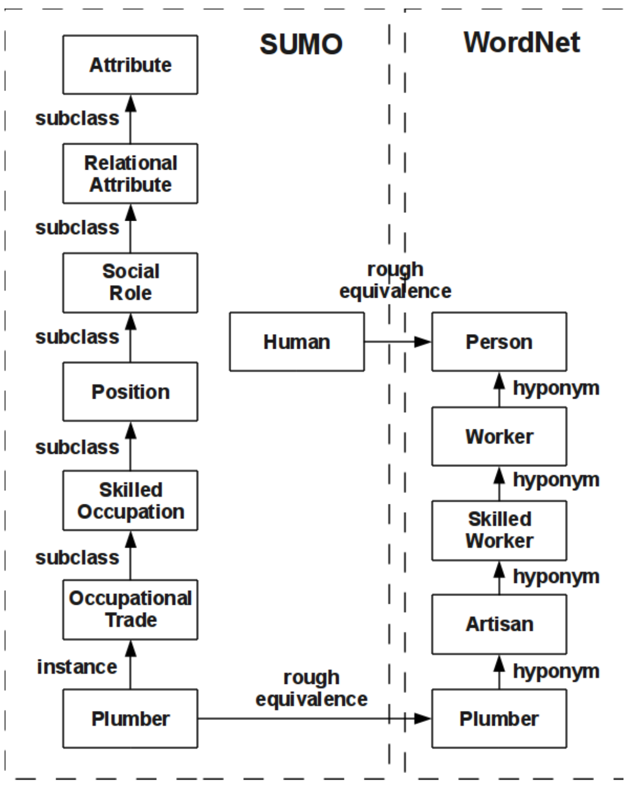


Figure 3: Comparing hierarchies of SUMO and WordNet

While this paper is not principally about the SUMO-WordNet mappings, it is worth stating at least briefly that the two products have very different roles. SUMO is a formal ontology, stated in a particular mathematical logic with associated inference engines. It contains rules that allow it to be used in logical deduction. SUMO is – to the best of the ability of current theorem provers to determine – logically consistent, and we continuously strive to find and eliminate inconsistencies with improved theorem proving technology. SUMO is a product constructed intentionally by humans. New terms and definitions can be added at will to model reality and do not need to mirror the presence (or absence) of linguistic tokens in any human language.

In contrast, WordNet is a lexical database of English. Lexical tokens are collected from the use of English and may not be arbitrarily created by the WordNet developers. Lexical tokens are not formal, mathematical entities. Words can be vague and ambiguous. Many words cannot be given formal definitions. The semantic relations of WordNet are not necessarily truth preserving through an arbitrary number of links. With a very small number of available semantic relations a number of logical notions are necessarily conflated. For example (see Figure 3), in WordNet, "plumber" is a type of "human" despite that being a transient role that is not true throughout the life of any plumber, whereas "ape" is a type of "primate" and that fact is indeed true throughout the life of any ape. 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.

The "role" relation (appropriate for relating individuals and kinds of jobs) is different from the "type" relation (relating a general class of things to a more specific class of things). One might argue that WordNet should simply add a new semantic relation of "role". However, there are thousands of such relations. Do all of them get added to WordNet?

We should note that this example, and many others that could be cited are not criticisms of WordNet. WordNet is designed to represent language, not a logically consistent reality. The "hyponym/hypernym" relation is intended to represent linguistic notions, especially the "substitution test" which allows more general words to be substituted for more specific words in a sentence without making a sentence nonsensical.

Most importantly, SUMO has an entirely open set of statements that can be made involving multiple concepts. Where the set of WordNet semantic relations is limited to just a fixed and small set of binary relations that are linguistically justifiable, SUMO has an open set of thousands of relations, and rules which combine sets of arbitrary numbers of terms in complex and productive ways that are capable of expressing the full set of facts that govern our reality. WordNet is appropriate for modelling language. SUMO is appropriate for modelling truths about the world.

Having SUMO and WordNet as distinct but linked products allows us to separate language and logic and not have linguistic concerns impact the representation of reality, or the goal of representing the world disturb the accurate representation of human language as written and spoken. Having linked these different resources allows us a rich basis for understanding language [33].

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.

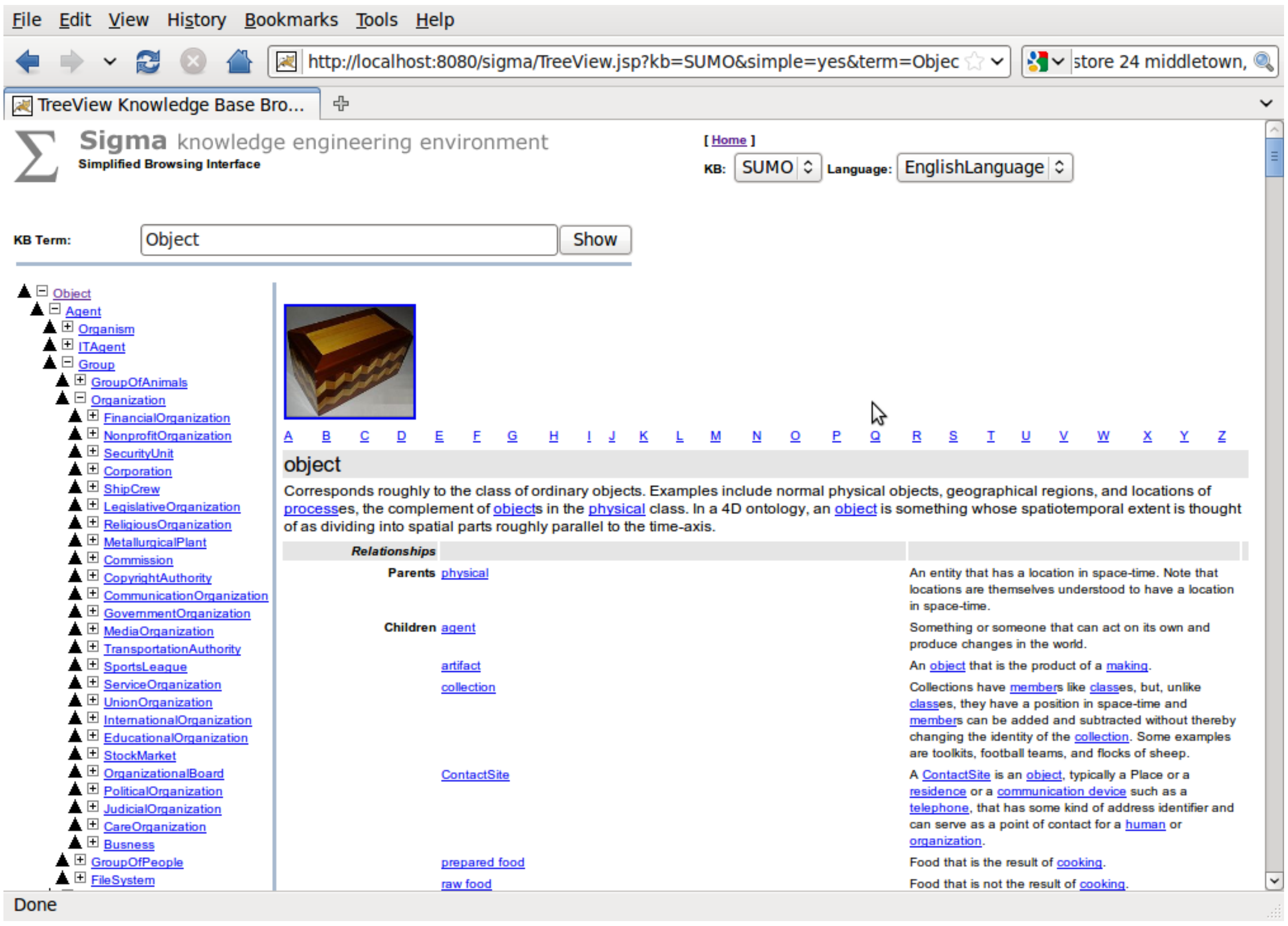
4 BROWSING and DISPLAY

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 2). 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. The mechanism in Sigma for language generation is simple, but with a very large ontology used as the source of language generation the richness and coverage of the resulting statements is still significant. 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 (Figure 4) 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.

In 2007-2008 a significant effort was undertaken to find open source images that could be linked to provide an informal visual representation of as many of the concepts in SUMO as possible. Some 12,000 links were made by hand to public domain icons and images in Wikipedia. In 2009 Princeton University published results of a project to link images to WordNet.


File Edit View History Bookmarks Tools Help
http://localhost:8080/sigma/TreeView.jsp?kb=SUMO&simple=yes&term=Objec
store 24 middletown,
TreeView Knowledge Base Bro...
Sigma knowledge engineering environment
Simplified Browsing Interface
[Home]
KB: SUMO
Language: EnglishLanguage
KB Term: Object
Show
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Done

Object
- Agent
  - Organism
  - ITAgent
  - Group
    - GroupOfAnimals
    - Organization
      - FinancialOrganization
      - NonprofitOrganization
      - SecurityUnit
      - Corporation
      - ShipCrew
      - LegislativeOrganization
      - ReligiousOrganization
      - MetallurgicalPlant
      - Commission
      - CopyrightAuthority
      - CommunicationOrganization
      - GovernmentOrganization
      - MediaOrganization
      - TransportationAuthority
      - SportsLeague
      - ServiceOrganization
      - UnionOrganization
      - InternationalOrganization
      - EducationalOrganization
      - StockMarket
      - OrganizationalBoard
      - PoliticalOrganization
      - JudicialOrganization
      - CareOrganization
      - Busness
    - GroupOfPeople
    - FileSystem

object

Corresponds roughly to the class of ordinary objects. Examples include normal physical objects, geographical regions, and locations of processes, the complement of objects in the physical class. In a 4D ontology, an object is something whose spatiotemporal extent is thought of as dividing into spatial parts roughly parallel to the time-axis.

Relationships		
Parents	physical	An entity that has a location in space-time. Note that locations are themselves understood to have a location in space-time.
Children	agent	Something or someone that can act on its own and produce changes in the world.
	artifact	An object that is the product of a making.
	collection	Collections have members like classes, but, unlike classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the collection. Some examples are toolkits, football teams, and flocks of sheep.
	ContactSite	A ContactSite is an object, typically a Place or a residence or a communication device such as a telephone, that has some kind of address identifier and can serve as a point of contact for a human or organization.
	prepared food	Food that is the result of cooking.
	raw food	Food that is not the result of cooking.

Figure 4: Simplified browser view

Although the links are public, many of the images in their corpus do not have an open license. We looked only at those images linked to WordNet synsets that have a rough equivalence mapping to SUMO terms. As a result of both restrictions, only 900 images which were linked to Wikipedia in Princeton's ImageNet [28] corpus were imported.

5 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 formal reasoning. However, in expressive ontologies, such as SUMO, we can generally not expect that all contradictions can be detected with theorem provers or that consistency can be formally proved (note, for example, that Peano arithmetic can be formalized in SUO-KIF). But that does not rule out that such goals can nevertheless be achieved for many concrete theories, in particular, such theories which do not make use of the full expressive power of SUO-KIF. Sigma therefore provides different tools for quality checking, combining exhaustive and terminating special purpose tests with incomplete and generally non-terminating general purpose testing based on theorem proving or model finding.

Moreover, a large theory may be inconsistent while still being used for practical theorem proving and question answering. It is not desirable, but just a fact of life. A theory is either consistent or not, but just because a theory potentially contains some hidden inconsistency, this does not mean that this inconsistency will influence any given proof in practical applications of the ontology (if so, then it is also more likely that the inconsistency can in fact be detected and eliminated in the first place by theorem proving and model finding means). And even if this happens in rare cases, then there is still the possibility to check the delivered proof or argument by hand and to reject it based on this a posteriori verification. Inconsistencies may theoretically linger undetected for years and may never become practically relevant.

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.

There are two special case tests for errors that must be corrected. 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 a term has 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 ultimate 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 classes or relations, 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.

A further test exploits the SUMO-WordNet mappings. They offer the opportunity to find problems exposed by differences in the two products. As discussed above, we believe that the two hierarchies should not necessarily be isomorphic, and therefore respective differences do not necessarily mark an error.

In the diagnostics 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. An example of a mismatch in the two hierarchies is shown in Figure 3.

6 INFERENCE

Sigma can be used as a whole for theory development, employing its inference component in the service of testing and debugging a theory. The inference portion of Sigma can also be used as an embeddable component in applications involving reasoning. An example application, the generation of stories for small children, will be discussed in detail in section 11 below.

The inference interface of Sigma consists primarily of two Java methods: *ask*, and *tell*. Clients tell statements in SUO-KIF to the knowledge base and then ask queries in SUO-KIF (along with some performance parameters such as the amount of time allowed for finding an answer). The result of tell, if an answer is found, is a binding for any free variable in the query, along with a formal proof of how the answer was determined. In the story generation application, those bindings are then used to construct other queries or assertions.

Since 2003, Sigma has used an open-source, customized version of the Vampire [29] theorem prover called KIF-Vampire. Our experience with practical knowledge base systems has shown that there are several features that are often needed, yet also usually absent from high performance theorem provers. In a typical use case in decision support applications, a user wants to be able to pose many queries to a knowledge base where most of the knowledge does not change from query to query, and where the set of available knowledge is quite large. The user expects to get an answer to a query, to be able to specify a timeout for difficult queries, and request multiple answers to the same query, if available. The user expects to get some information that justifies why an answer is true. Performing basic arithmetic as part of inference is also desirable and even necessary for many common sense inferences. Each of these features needed to be added to Vampire by researchers at University of Manchester, to create the new 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. The same transformations are needed for the TPTPWorld interface, along with an additional set of transforms (Figure 5). We will first discuss the pre-processing steps and then address post-processing

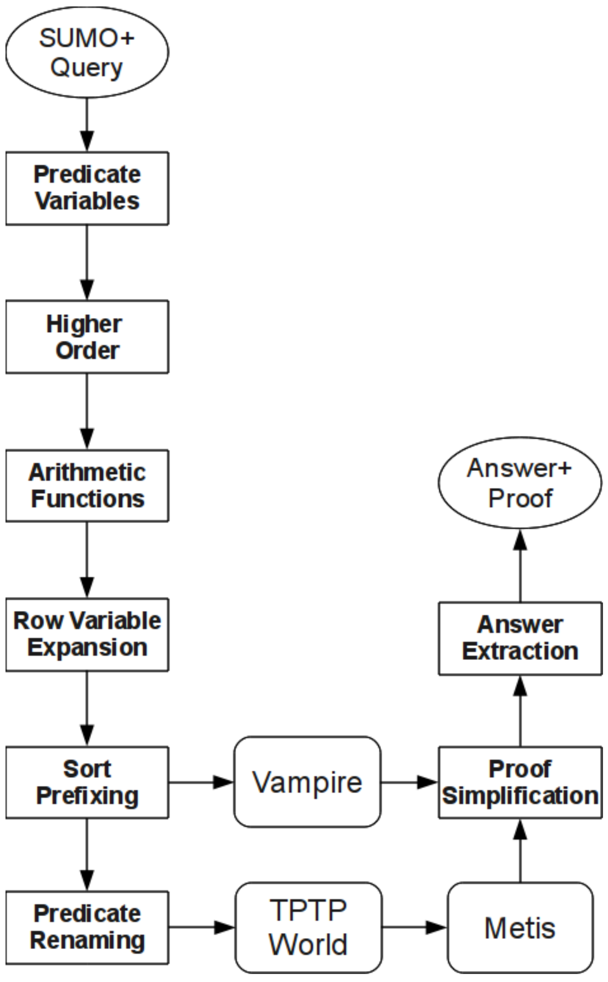


Figure 5: Sigma pre- and post-processing steps

We have implemented two approaches for the first step of removing variables from the predicate position. Our first approach was to add a "dummy" predicate to all clauses other than those with logical operators. For example, the following axioms,

```
(instance part TransitiveRelation)

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

become

```
(holds instance part TransitiveRelation)

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

This however resulted in worse performance for theorem 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. The rule above will be duplicated with the variable **`?REL`** being instantiated with every **`TransitiveRelation`** as in

```
(=>
  (and
    (part ?INST1 ?INST2)
    (part ?INST2 ?INST3))
  (part ?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 the next preprocessing step we turn embedded formulas into uninterpreted lists of symbols by quoting them. 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.

For example,

```
(believes John (likes Mary Jeff))
```

becomes

```
(believe John `(likes Mary Jeff))
```

This allows KIF-Vampire to perform very simple queries on higher order statements, such as

```
(believes John `(likes Mary ?X))
```

and get the correct answer of **`Jeff`**. However, logical symbols in the embedded formulas lose their meaning, so if

```
(believes John
  '(and
     (likes Mary Jeff)
     (likes Bill Sue)))
```

is asserted, the same query will fail, as the **`and`** does not have its conventional meaning, and the two lists will not unify.

In a simple third step, Sigma translates SUMO basic arithmetic functions into the native symbols required by KIF-Vampire. For the fourth preprocessing step 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. This macro expansion approach does change the semantics of row variables, simplifying the logic and improving its computational properties. This limitation however does not appear to have adverse practical consequences for common-sense knowledge representation, which is the goal of SUMO.

To explain what is done, take the following example, where the axiom

```
(=>
  (and
    (subrelation ?REL1 ?REL2)
    (?REL1 @ROW))
  (?REL2 @ROW))
```

becomes

```
(=>
  (and
    (subrelation ?REL1 ?REL2)
    (?REL1 ?ROW1))
  (?REL2 ?ROW1))
```

and

```
(=>
  (and
    (subrelation ?REL1 ?REL2)
    (?REL1 ?ROW1 ?ROW2))
  (?REL2 ?ROW1 ?ROW2))
```

etc. up to the maximum arity currently allowed of 7. Note that in axioms such as this, which also require predicate variable instantiation, we must restrain the expansion to only those arities which are compatible with the instantiated predicates. For example, **`located`** is a **`subrelation`** of **`partlyLocated`** and both have arity 2. So, **`@ROW`** will only be expanded to the case of two variables. In the few cases where axioms have two row variables, this can result in 49 new axioms.

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 the University of Miami to remotely 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, this 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 5th step to the Sigma 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.

For example, consider the rule

```
(=>
  (and
    (instance ?TRANSFER Transfer)
    (agent ?TRANSFER ?AGENT)
    (patient ?TRANSFER ?PATIENT))
 (not
    (equal ?AGENT ?PATIENT)))
```

All relations in SUMO are typed. While we have an explicit type stated for **`?TRANSFER`**, none is given in the rule for **`?AGENT`** and **`?PATIENT`**. However, we know from the definitions of agent and patient that their second arguments are given respectively as

```
(domain agent 2 Agent)
(domain patient 2 Object)
```

We can then modify the rule to add a new precondition with those type restrictions.

```
(=>
  (and
    (instance ?AGENT Agent)
    (instance ?PATIENT Object))
  (=>
    (and
      (instance ?TRANSFER Transfer)
      (agent ?TRANSFER ?AGENT)
      (patient ?TRANSFER ?PATIENT))
    (not
      (equal ?AGENT ?PATIENT))
```

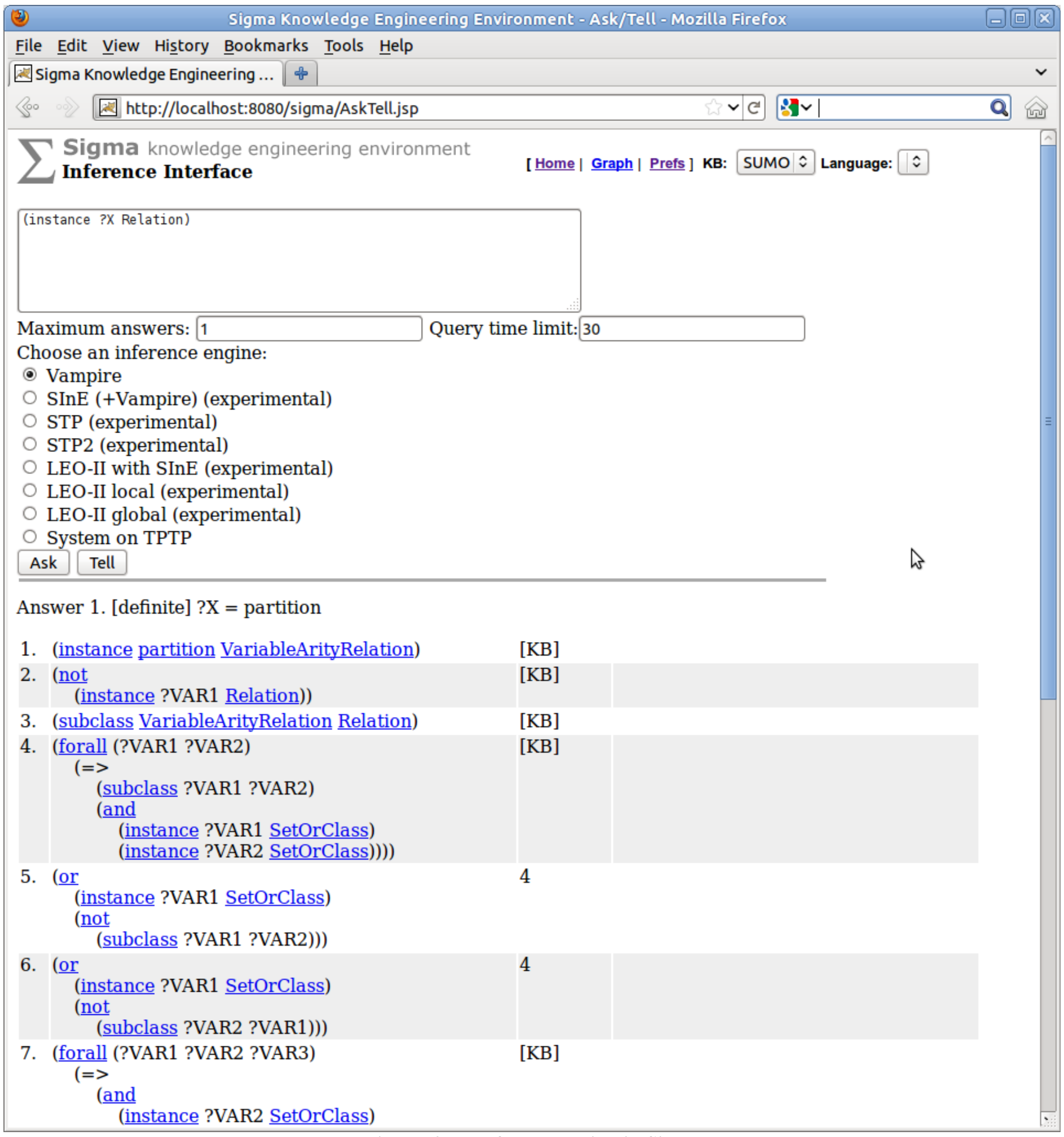
Combining these different preprocessing operations with the capability to generate TPTP language versions of SUMO allowed us to use SUMO-based tests in the yearly CASC competition [14,15], stretching theorem prover developers to work on high performance results in a new category of problems in which 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.

In addition to preprocessing, some post-processing is needed for all theorem provers that are used in Sigma. All the TPTP provers that report full proofs, as well as KIF-Vampire, present and ordered list of deductions, where premises are given and then a conclusion. In presenting a proof to the user (Figure 6), we would like to avoid showing the same axiom many times if it is used in several proof steps. We therefore assign a numerical index to each axiom, in order of its appearance in the proof. The indexes can then be referenced when they are preconditions to a listed step, making the proof appear more similar to what a logic student will be used to from a standard textbook presentation of a proof.

An answer variable is a binding for a variable in a query that is unbound. In the proof shown in Figure 6 ?X is an unbound variable in the query. For TPTP systems that do not report answer variables, or handle more than one answer per query, a more complicated approach is needed. For systems such as EP, that report proofs but not answer variables, the axioms in the proof are resubmitted to the Metis prover [30] which does report answer variables. Multiple answers are found by resubmitting the query with a new clause added that excludes previous answers. For the example query shown, in order to get the second answer, the new query would become

```
(and
  (instance ?X PrimaryColor)
  (not
    (equals ?X Red)))
```

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 in SUMO, Sigma includes a consistency check function that leads the theorem prover to consider each axiom in a knowledge base. This is an important point because a user may have a knowledge base that is inconsistent, but in practice may make many useful inferences over a long period of time while never having the problem show up in a proof. For example, take the knowledge base


Sigma Knowledge Engineering Environment - Ask/Tell - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Sigma Knowledge Engineering ...
http://localhost:8080/sigma/AskTell.jsp
Sigma knowledge engineering environment
Inference Interface
[Home | Graph | Prefs] KB: SUMO Language:
Ask
Tell

(instance ?X Relation)

Maximum answers: 1 Query time limit: 30

Choose an inference engine:

- [✓ Vampire]
- SInE (+Vampire) (experimental)
- STP (experimental)
- STP2 (experimental)
- LEO-II with SInE (experimental)
- LEO-II local (experimental)
- LEO-II global (experimental)
- System on TPTP

Answer 1. [definite] ?X = partition

```
1.  (instance partition VariableArityRelation)      [KB]
2.  (not                                            [KB]
      (instance ?VAR1 Relation))
3.  (subclass VariableArityRelation Relation)       [KB]
4.  (forall (?VAR1 ?VAR2)                           [KB]
      (=>
        (subclass ?VAR1 ?VAR2)
        (and
          (instance ?VAR1 SetOrClass)
          (instance ?VAR2 SetOrClass))))
5.  (or                                             4
      (instance ?VAR1 SetOrClass)
      (not
        (subclass ?VAR1 ?VAR2)))
6.  (or                                             4
      (instance ?VAR1 SetOrClass)
      (not
        (subclass ?VAR2 ?VAR1)))
7.  (forall (?VAR1 ?VAR2 ?VAR3)                     [KB]
      (=>
        (and
          (instance ?VAR2 SetOrClass)
```

Figure 6: Proof presentation in Sigma

```
(fatherOf John Bill)
(fatherOf John Mark)
(=>
  (fatherOf ?X ?Y)
  (and
    (not
      (exists (?Z)
        (and
          (fatherOf ?X ?Z)
          (not
             (equal ?Y ?Z)))))))
```

While this knowledge base is trivially small, imagine that there are tens or hundreds of thousands of other statements, many of which may involve the predicate symbol **fatherOf**. A complete examination of the proof space is impossible. Imagine that the user poses the query

```
(fatherOf John Bill)
```

Given a finite and small amount of time with which to find an answer, the prover may just find and return "yes" after encountering the first assertion. It may not continue the search process to find the contradiction. In fact, given a large and complex enough knowledge base, and a complex enough contradiction, it might not be found for years.

To help guide the search for contradictions, Sigma takes each axiom, which is loaded one by one starting with an empty knowledge base. For each axiom, the prover is asked to compute 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 (although the answer also may not be nonsensical, as we have explained, so this is a conservative approach).

Once processing finishes, redundancies are collected and reported. At its simplest, a redundancy can be a duplicated statement, and that is clearly an error. Although initially harmless, having the same statement in two places can easily lead to problems as an ontology evolves, as one statement might get changed while a duplicate does not. For example, a developer might forget that a domain ontology file already has a statement

```
(subclass Table Furniture)
```

and assert the same statement in a different file.

A more complex case is where one statement is simply deducible from several others. This is often intentional, as knowledge engineers may wish to short-circuit a common chain of reasoning in order to have faster inference. Such a case is even more likely to suffer from the problem of changes not being reflected in the chain of deductions, and the redundant conclusion. For example,

```
(=>
  (instance ?P TransitiveRelation)
  (=>
    (and
       (?P ?A ?B)
       (?P ?B ?C))
    (?P ?A ?C)))

(subclass Table Furniture)

(subclass DiningTable Table)

(subclass DiningTable Furniture)

(instance subclass TransitiveRelation)
```

An intriguing possibility in contradiction detection would be to continue processing, knowing that the theorem prover may not run across the knowledge needed to prove a contradiction for a different query. We might also explore treating an inconsistent knowledge base in a four-valued logic, where each axiom can be provably true, false, both or one or the other [31]. We might also explore whether an automatic process can be created to remove random axioms from a proof of contradiction, checking to see whether a contradiction can still be found, and reporting the deleted axiom to the user when it is not. This may assist the user in determining the appropriate correction to make by finding a subset of the axioms in the proof of contradiction that appear to be most responsible.

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.

7 HIGHER ORDER LOGIC

Another recent innovation is in translating SUMO to the new typed higher order format TPTP THF [18] for use by true higher order theorem provers [17,44]. The goal of this work is to better support higher order aspects in SUMO, in particular, embedded formulas, temporal operators such as "**holdsDuring**" and epistemic operators like "**knows**" and "**believes**". The first-order based support for these concepts in Sigma is limited, with the effect that many desirable inferences are not supported, certain queries cannot be answered, and some potential inconsistencies cannot be detected. The following example on reasoning within temporal contexts illustrates the challenge. It expresses that whatever holds, holds at all times, that Mary likes Bill, and that during 2009 Sue liked whoever Mary liked.

```
(=>
  ?P
  (holdsDuring ?Y ?P))

(likes Mary Bill)

(holdsDuring
  (YearFn 2009)
  (forall (?X)
    (=>
      (likes Mary ?X)
      (likes Sue ?X))))
```

A higher order theorem prover such as LEO-II [35], which has been integrated into Sigma, can now effectively (in about a tenth of a second) answer, for this knowledge base, queries like whether Sue liked Bill in 2009.

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

or whether there is a year in which Sue has liked somebody.

```
(holdsDuring
  (YearFn ?Y)
  (likes Sue ?X))
```

The rule **(=> ?P (holdsDuring ?Y ?P))** can also be replaced by **(holdsDuring ?Y True)**, and LEO-II finds an answer even more quickly. At the same time, this example is out of reach of the first order reasoning techniques described above.

A key aspect in the solution of the example is Boolean extensionality, which ensures that the denotation of each formula, and also of the embedded ones, is either true or false. This assumption has actually never been questioned for SUMO. However, assuming Boolean extensionality also leads to problematic effects as the following slight modification of the example illustrates (instead of the temporal context we now consider an epistemic context).

```
(knows ?Y True)

(likes Mary Bill)

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

It is not a surprise that, given this knowledge base instead of the previous one and by using a similar reasoning pattern as before, LEO-II can effectively confirm the query

```
(knows Ben (likes Sue Bill))
```

However, this inference is disturbing since we have not explicitly required that `(knows Ben (likes Mary Bill))` holds, which intuitively seems mandatory.[2]

Our example illustrates, that modalities have to be treated with great care in classical, extensional logic. Our ongoing work therefore studies how we can suitably adapt the modeling of affected modalities in SUMO in order to appropriately address this issue.

The solution we currently explore is to map SUMO reasoning problems that involve modal operators to problems in quantified multi-modal logics. Unfortunately there are only very few direct theorem provers for quantified multimodal logics available. We therefore exploit our recent embedding of quantified multimodal logics in classical higher order logic [39] and we investigate whether this embedding can fruitfully support the automation of modal operators in SUMO with off-the-shelf higher order automated theorem provers.

Our ongoing research studies how non-classical reasoning can generally be integrated with and realized in classical higher order logic and how higher order theorem provers and model finders can be utilized for the task. So far we have studied propositional modal logics and propositional intuitionistic logics [36], access control logics [37], quantified modal logics [39], and conditional logics [40]. Most importantly, combinations of these logics can be achieved in classical higher order logic [38], which is what we ultimately need in order to address challenge interactions of modal operators in SUMO.

9 MAPPING, MERGING and TRANSLATION

In addition to SUO-KIF and TPTP syntax, 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 more expressive 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], among others. Such an alignment is problematic however, because little verification is possible. As is typical of most products that are being called ontologies, OBO consists mostly of taxonomic relations, with no rules and few axioms besides class membership.

10 WORKING with SIGMA and SUMO

There are as many possible processes for formal ontology development as there are for software development. Small projects may benefit from the low overhead of an informal process. Large projects with big teams will benefit from a greater degree of formal process. A typical process employing Sigma to extend SUMO is as follows:

[2]It is important to note that `True` in A' can actually be replaced by other tautologies, e.g. by `(equal Mary Mary)`.

- Developers use a set of instructions or documents as a source, or write down text in natural language that describes the domain of interest.

- The text is used as a basis for creating a glossary of natural language terms and definitions

- Developers examine the SUMO hierarchy (using the term browser, and tree/graph browser) for each term in the glossary. The WordNet search pages are used to find all the different meanings of each defined word in the source text, and the WordNet-SUMO mappings are used to find the formal SUMO term that best fits the intended meaning of the textual term. For any substantially new and specialized domain, the task is to find a more general term that encompasses the meaning of the more specific textual term. Textual terms that are already covered by specific SUMO definitions are put aside as complete. For new terms and definitions that are needed, developers begin by adding subclass or instance statements to the appropriate leaf term in SUMO, by creating and editing a text file in SUO-KIF format.

- Once a preliminary SUO-KIF file has been created, developers load it into Sigma, along with SUMO proper and all the other domain ontologies the new file may extend. Developers run the Sigma Diagnostics to find any errors.

- Developers use the information in the natural language definitions created earlier to guide creation of SUO-KIF axioms. Each class should have at least a subclass statement and a documentation statement. Each relation should have domain statements defining the class membership of its arguments, and be defined as an appropriate type of relation, such as **`TransitiveRelation`**. Each term should have at least one rule, that helps to make the term usable for inference. If there are very few things that can be stated about the term, reconsider whether it should be created.

- Developers create **`format`** and **`termFormat`** statements in the language of choice to support natural language paraphrases in Sigma for the axioms previously written. These can be presented to domain experts to help confirm that the desired knowledge has been captured correctly.

- Developers map the terms in the ontology to WordNet. This is accomplished by placing links in the existing SUMO-WordNet mapping files (if mapping to English) that update the existing links where needed to point to the more specific terms that have just been created

- Developers load the revised WordNet mapping files into Sigma and use the Sigma WordNet Diagnostics to see where the WordNet hierarchy may differ from the formal relationships created in the new ontology. The existence of differences is not necessarily bad, but they should be examined and understood.

- Developers run the Sigma Consistency Check to find any logical contradictions in the new theory. Normally, there will be many cycles of adding content, then running the Diagnostics and Consistency Check processes in Sigma to find and correct errors as the theory is elaborated. At each iteration where no errors are found, in a group development process, the theory would be uploaded to a source configuration management system such as CVS or Subversion. Other developers are then free to test new theories with respect to their own work, and coordinate with each other. One can view the Diagnostics and Consistency Check steps as analogous to compilation and build of a conventional procedural computer program.

- Peer review is one of the best ways to improve a theory. Sigma helps developers significantly beyond just reviewing a file of declarative code, allowing them to search and test a theory in many different ways.

11 EXAMPLE APPLICATION

The primary use of Sigma has been as an IDE for ontology, rather than as an embeddable component. Many ontologies are developed as an end in themselves for structuring information or supporting database design. An example of this sort of application is [62].

In order to provide an example of how Sigma is used in practice to develop embedded applications, we now discuss the "SUMO Stories" project, which uses SUMO and Sigma to automatically develop short stories for small children. Elsewhere [34] we discuss the project in more detail. Here we will focus on how Sigma supports the project.

The first step was to collect the knowledge relevant to the application and state it as informal English sentences. In this case, a prior application [41] that used linguistic methods for story production rather than deductive inference, provided this corpus of sentences. An example fragment of a story from [42] is

"The afternoon was windy. Rizzy the rabbit was in the dining room. She played near a lamp. Rizzy broke the lamp. She was scared. "

In simple sentences from a children's story, it is easy to pick out the concepts that need to be captured formally - "afternoon", "windy", "dining room", "lamp", "breaking", and "scared". An experienced user of SUMO may already know the names for the likely SUMO classes that encode these concepts or their parents and be able simply to enter names and the "KB term" field of the browser (see Figure 2), then confirm that the definitions match the user's intuitions about the way the words are used in the text. A less experienced user will enter words in the "English Word" field to search WordNet word senses, and see the SUMO links for each word sense.

For one example term, take "breaking". It has 59 different senses in WordNet. The most appropriate senses are linked to SUMO's **`Damaging`** and **`Destruction`**. The difference in those two subclasses of process are a case of extent, with **`Destruction`** being the more serious. By the formal axiom (and the definitions of the other terms that appear in the axiom)

```
(<=>
  (instance ?PROCESS Destruction)
  (exists (?PATIENT)
    (and
      (patient ?PROCESS ?PATIENT)
      (time ?PATIENT
        (BeginFn
          (WhenFn ?PROCESS)))
      (not
        (time ?PATIENT
          (EndFn
            (WhenFn ?PROCESS)))))))
```

we see that **Destruction** entails that the patient of the process or some essential part of it must cease to exist at the end of the event. In the case of a child breaking a lamp, the lamp might simply have a crack in its base. We can only state a more vague and general notion of breaking equivalent to the SUMO notion of **Damaging**.

We should note that the formal axiom makes clear the differentiation between the two classes. If we had only a hierarchical relationship specifying that one term was a specialization of the other, or even the informal English description of the terms that is also present in SUMO, we would not be able to make this informed choice, or have the consequent of the rule above follow as a logical fact from asserting the existence of a **Destruction**.

In any application, there are likely to be notions that are not already completely captured by existing SUMO terms. Human emotions comprise one such area. Formalizing such a complex area was outside the scope of the SUMO Stories effort, but simple subclasses of the existing **EmotionalState** attribute class were created to handle notions such as being "scared".

Rules were defined to govern the behavior of the story characters. This proved to be challenging, since rules needed to state that certain actions were possible or likely, but in many situations these were not definite consequences of certain states in the story world. A child playing near a breakable lamp in the story likely results in a broken lamp, but not always. Also, a story generator that generates the same story for the same set of initial conditions each time is not very interesting. We began with statements employing the SUMO **capability** relation that provides an embedding of a modal statement inside a first order logic (we will return to explain why this is a modal in a few paragraphs below). However, it proved too limiting for the story description. We are now exploring a lightweight implementation of a probabilistic logic framework, but still using the existing SUMO terms.

In the current implementation, we created rules that express what the characters are capable of doing given the environmental conditions and phase of the story. Java code calls Sigma's inference API with a query template such as

```
(capability ?P ?R ?O)
```

where the **capability** relation takes three arguments: a process type, a role that a thing may play in a process, and an instance of an Object. There may be several answers to such a query. One might be that

```
(capability
  RecreationOrExercise
  experiencer
  Rizzy)
```

meaning that Rizzy the rabbit character is capable of playing, given the current state of the world.

We should digress now for a moment to explain why the capability relation is considered modal. Given the challenges involved in true higher order reasoning, we have also tried to provide first order encodings in SUMO for notions that traditionally would require a higher order statement. The notion of **capability** is one such expression. The relation is used to state that events of a certain type and relation to another object may occur. A more traditional encoding of the statement above would be

```
(possible
  (exists (?R)
   (and
     (instance ?R RecreationOrExercise)
      (experiencer ?R Rizzy))))
```

Such statements are fairly common, and yet a higher order encoding can be a barrier to successful practical inference. A first order encoding sacrifices some of the semantics of the higher order version, but enough utility is retained in practice to make statements of this type a valuable addition to SUMO.

Returning now to our example, answers are returned in a prescribed format, with successful queries resulting in either a yes/no answer for queries with no variables, or in an answer structure that provides bindings for variables (such as "**?R = experiencer**" in the example above), accompanied by a proof of how those binding were found. SUMO's Java process will pick one of the capabilities to assert (again through Sigma's inference API) as a true fact in the story world. SUMO's Java process will then ask another query to determine the events in the next phase of the story.

12 Related Work

Numerous ontology editors and knowledge engineering tools exist today. Prominent examples include Protégé [47], Specware [48], SWOOP [49], Top Braid composer [54], OilED [50], WebODE [51], Ontolingua [52], KAON [53], Internet Business Logic [55], OntoTrack [56], SemanticWorks Semantic Web tool [57] and IHMC Cmap Ontology Editor [58]. Survey articles exist that compare and summarize main features of such tools [59,60,61]. These surveys show that actually very few editors and tools exist that support expressive languages such as CycL, KIF, or SUO-KIF.

A particular unique feature of Sigma is that it is directly linked to the TPTPworld [12] infrastructure and that it integrates various off-the-shelf first-order automated theorem provers and very recently even an off-the-shelf higher-order theorem prover. In this regard, the largest body of work potentially related to Sigma is instead actually part of the system already. These integrated reasoners support consistency checking but they can also be applied for other purposes including, for example, question answering. It should be possible to enhance tools like Protege also with some of the reasoning functionality supported in Sigma. However, this would clearly require major implementation effort. With regard to integration of theorem provers the Specware system is probably closest related to Sigma.

In many respects Sigma is also related to the ProofGeneral system [26]. ProofGeneral provides powerful and configurable interfaces which help user-interaction with proof assistants. In Sigma we are mainly interested in interfaces to automated theorem provers though. However, like in ProofGeneral, our interest is to support the presentation and explanation of machine proofs to the user.

13 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.

There are several efforts we are pursuing to expand the functionality and utility of Sigma.

First is to package a formal release. It has proven difficult to provide regular binary releases with consistent functionality and documentation at regular intervals, even though open source development releases have always been available to those who are willing to compile from source and deal with a research toolkit.

Second is a major effort to provide a formal semantics for SUO-KIF's higher order statements in combination with choosing an appropriate semantics for SUMO's modal operators, and to accordingly adapt our recent translation to THF [18]. LEO-II and other THF compliant provers can then be uniformly applied to problems encoded in SUO-KIF, including the SUMO, and they can subsequently be improved with regard to the particular challenges in question answering.

Once a formal semantics for SUO-KIF's higher order statements is fixed, we ideally would also like to devise a highly trusted verification component for SUMO. The idea is that this verification component should be capable of exploiting proof information (such as generated answer to queries and axiom selections) from other inference engines in order to reconstruct proof objects on its own in a highly trusted (ideally fully verified) reasoning engine. The hope is that with additional proof information available reconstruction of the proofs should be generally possible even when the trusted reasoning engine is not particularly well suited for high performance proof automation. Overall we would thereby achieve a two level support for proof automation in SUMO: the aim of the first level would be to provide various means for high performance automated proof search, and verifying the soundness of the generated answers would be the task of the second level. Consulting the second level could be optional, for example, only if a user finds a query answer suspicious. In safety critical applications it could be made a general standard though. The advantage of the two level approach would be that even potentially unsound proof automation means could be considered as useful in Sigma, provided that their soundness leaps are only of low practical relevance. These few soundness leaps could then still be detected by the second level.

Lastly, we are developing a simple first order theorem prover in Java as an integrated part of Sigma. This will serve as an educational tool, and a testbed for developing features for theorem provers that demand access to the internals of a prover.

Acknowledgements

Sigma has been a decade-long group effort. Due to space constraints it is impossible to mention all contributors. Readers are encouraged to recognize all co-authors on the referenced papers as contributors, as well as those who have written the actual Java code in Sigma, and who are credited in the code itself at the SourceForge CVS logs [43]. Besides the authors of this paper, some major contributors include Michal Sevcenko, Frank Theiss, Krystof Hoder, Steven Trac, John Li, Nick Siegel, Geoff Sutcliffe, Ian Niles and Randall Schulz.

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 Proceedings.
[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.), pp. 2-9.
[4] de Melo, G., Suchanek, F., and Pease, A., (2008). Integrating YAGO into the Suggested Upper Merged Ontology. In Proc. 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. In 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. In 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 Proc. of the CADE-21 workshop on Empirically Successful Automated Reasoning on Large Theories (ESARLT).
[14] Sutcliffe, G., (2009). The TPTP Problem Library and Associated Infrastructure. Journal of Automated Reasoning, 43(4):337-362..
[15] Pease, A., Sutcliffe, G., Siegel, N., and Trac, S., (2010). Large Theory Reasoning with SUMO at CASC. Special issue on Practical Aspects of Automated Reasoning, AI Comm., 23(2-3):137-144. IOS Press.
[16] Hoder, K. (2008) Automated Reasoning in Large Knowledge Bases, PhD thesis, Charles University, Prague, Czech Republic.
[17] Benzmüller, C., and Pease., A., (2010). Progress in Automating Higher Order Ontology Reasoning. In Proceedings of the Second

International Workshop on Practical Aspects of Automated Reasoning, Konev,B., Schmidt, R.A., and Schulz, S., (eds.).

[18] Sutcliffe, G., and Benzmüller, C., (2010) Automated Reasoning in Higher Order Logic using the TPTP THF Infrastructure. Journal of Formalized Reasoning, 3(1):1-27.

[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] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L., Dean, M., Schreiber, G., (Ed.) (2004). OWL Web Ontology Language Reference, World Wide Web Consortium, Recommendation, Feb. 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 (eds.), August 16-17, Lisbon, Portugal.

[23] Li, J., (2004) LOM: A Lexicon-based Ontology Mapping Tool, in Proc. of the Performance Metrics for Intelligent Sys. conf. (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., (2006). Ontology Development Tools for Ontology-Based Knowledge Management. Encyclopedia of E-Commerce, E-Government and Mobile Commerce, Idea Group Inc.

[26] Aspinall, D., (2000). Proof General: A Generic Tool for Proof Development. In Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of TACAS 2000, LNCS 1785. Springer, pp. 38-42.

[27] Fellbaum, C (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[28] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., (2009). ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR).

[29] Riazanov, A., and A. Voronkov, A., (2002). The Design and Implementation of Vampire. AI Communications, 15(2-3):91–110.

[30] Hurd, J, (2003). First-Order Proof Tactics in Higher-Order Logic Theorem Provers. In Archer, M., Di Vito, B., and Munoz, C., (eds.), Proceedings of the 1st International Workshop on Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports, pp. 56–68.

[31] Grant, J., and Hunter, A. (2008). Analysing inconsistent first-order knowledge bases. Artificial Intelligence 172:1064-1093.

[32] Schulz. S., (2002). E: A Brainiac Theorem Prover. AI Communications, 15(2-3):111.

[33] Pease, A., and Li, J. (2010) Controlled English to Logic Translation. In Theory and Applications of Ontology, ed. Roberto Poli, Michael Healy, and Achilles Kameas, Springer, ISBN: 978-90-481-8846-8.

[34] Cua, J., Manurung, R., Ong, E., and Pease, A., (2010). Representing Story Plans in SUMO, Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity, Los Angeles, CA, June 5, 2010.

[35] Benzmüller C., Paulson L.C., Theiss F., and Fietzke A., (2008). LEO-II - A Cooperative Automatic Theorem Prover for Higher-Order Logic. In Proceedings of the Fourth International Joint Conference on Automated Reasoning (IJCAR'08), LNAI volume. 5195, Springer, pp. 162-170.

[36] Benzmüller C. and Paulson L.C., (2010). Multimodal and Intuitionistic Logics in Simple Type Theory. The Logic Journal of the IGPL, 18(6): 881-892.

[37] Benzmüller, C., (2009). Automating Access Control Logic in Simple Type Theory via LEO-II. In Emerging Challenges for Security, Privacy and Trust, 24th IFIP TC 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18-20, 2009, Proceedings, IFIP vol. 297, Springer pp. 387-398.

[38] Benzmüller, C., (2010). Combining Logics in Simple Type Theory. In Proceedings of 11th International Workshop on Computational Logic in Multi-Agent Systems, LNAI 6245, Springer, pp. 33-48.

[39] Benzmüller, C., and Paulson L., (2011). Quantified Multimodal Logics in Simple Type Theory. Logica Universalis, to appear. For an earlier version see: C. Benzmüller and L. C. Paulson, Quantified Multimodal Logic in Simple Type Theory. Seki Report SR-2009-02 (ISSN 1437-4447), Saarland University, 2009.

[40] Benzmüller, C., Gabbay, D., Genovese, V., and Rispoli, D., (2011). Embedding and Automating Conditional Logics in Classical Higher Order Logic. Submitted.

[41] Solis, C., Siy, J.T., Tabirao, E., and Ong, E., (2009). Planning Author and Character Goals for Story Generation. Proceedings of the NAACL Human Language Technology 2009 Workshop on Computational Approaches to Linguistic Creativity, 63-70, Boulder, Colorado, USA.

[42] Hong, A., Solis, C., Siy, J.T., and Tabirao, E. 2008. Picture Books: Automated Story Generator. Undergraduate Thesis, De La Salle University, Manila, Philippines.

[43] Sigma SourceForge CVS web site http://sigmakee.cvs.sourceforge.net/sigmakee

[44] Benzmüller, C., and Pease, A., (2010). Reasoning with Embedded Formulas and Modalities in SUMO, 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 (eds.), August 16-17, Lisbon, Portugal.

[45] Rubin, D.L., Noy, Natalya F and Musen, M.A. "Protégé: A Tool for Managing and Using Terminology in Radiology Applications." Journal of Digital Imaging. J Digit Imaging (2007): 1-13

[46] Lenat, D., (1995). Cyc: A large-scale investment in knowledge infrastructure, Communications of the ACM 38-11 (Novemter).

[47] Protégé. The Prot ́eg ́e project, http://protege.stanford.edu , (2002)

[48] Specware. http://www.specware.org/

[49] Kalyanpur, Aditya., Parsia, Bijan., Hendler, James. : A Tool for Working with Web Ontologies In: Proceedings of the International Journal on Semantic Web and Information Systems, Vol.1, No.1, Jan-Mar (2005) (see also http://code.google.com/p/swoop/)

[50] Bechhofer, S., Horrocks, I., Goble, C., Stevens, R.: OILEd: a reasonable ontology editor for the semantic web In: KI2001, Joint German/Austrian conference on Artificial Intelligence, volume LNAI Vol. 2174, pages 396-408, Vienna (2001)

[51] Arpírez, J.C., Corcho, O., Fernández-López, M., Gómez-Pérez, A.: WebODE: a scalable worbench for ontological engineering. In: KCAP-01, Victoria, Canada (2001)

[52] Farquhar, A., Fikes, R., Rice, J.: The Ontolingua server: a tool for collaborative ontology construction. In: Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada (1996).

[53] KAON - The Karlsruhe Ontology and Semantic Web Tool Suite. http://kaon.semanticweb.org

[54] Top Braid Composer. http://www.topbraidcomposer.com

[55] Internet Business Logic. http://www.semanticweb.org/wiki/Internet_Business_Logic.

[56] Liebig, Thorsen., Noppens, Olaf.: OntoTrack: Fast Browsing and Easy Editing of Large Ontologies: In: Proceedings of the 2nd International Workshop on Evaluation of Ontologybased Tools (EON-2003) Sanibel Island, Florida, USA (2003)

[57] SemanticWorks Semantic Web tool. http://www.altova.com/semanticworks.html

[58] IHMC Cmap Ontology Editor. http://www.ihmc.us/groups/coe/

[59] Mike Bergman: The Sweet Compendium of Ontology Building Tools. http://www.mkbergman.com/862/the-sweet-compendium-of-ontology-building-tools/

[60] M. R. Khondoker, P. Mueller: Comparing Ontology Development Tools Based on an Online Survey, Proceedings of the World Congress on Engineering 2010 Vol I WCE 2010, June 30 - July 2, 2010, London, U.K.
[61] Michael Denny: Ontology Tools Survey, Revisited. http://www.xml.com/pub/a/2004/07/14/onto.html
[62] Pease, A., and Rust, G., (2008) Formal Ontology for Media Rights Transactions, in Semantic Web Methodologies for E-Business Applications, ed. Roberto Garcia. IGI publishing.