Sigma Knowledge Engineering Environment

User Guide

version 2/15/2004

written by Adam Pease
adampease @ earthlink . net

License

Pease, A., (2003). The Sigma Ontology Development Environment,
in Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems,
August 9, Acapulco, Mexico.

Bug reports, fixed code and new implemented features are welcome however, and will be shared with the Sigma user community.

Table of Contents

Installation

Follow the steps below to install Sigma

1. Install J2SE Java 2 SDK v1.4.1_05 [Windows (all languages, including English) version] (downloaded from http://java.sun.com/j2se/1.4.1/download.html). Other versions may work, but we don't guarrantee it.
2. Install Tomcat v4.0.4 (downloaded from http://archive.apache.org/dist/jakarta/tomcat-4/archive/v4.0.4/bin/jakarta-tomcat-4.0.4.exe). Other versions may work, but we don't guarrantee it.
3. Modify the startup script for Tomcat to use a large stack size. The Tomcat installation should have added an item "Apache Tomcat 4.0" to your Start->Programs menu. Beneath that menu should be an item "Start Tomcat". Right click on that menu item and select "Properties". Then modify the "Target" field to add the command line option "-Xmx300M". For example, depending on where you have installed the JDK, the beginning of the command should be similar to:

```
C:\j2sdk1.4.0_01\bin\java.exe -jar -Xmx300M
```

4. Copy the KBs directory from the CD to your Tomcat directory
5. Copy the vampire directory onto your hard drive. The inference engine executable is in `\vampire\Vampire_VSWorkspace\vampire\Release\kif.exe`
6. 6. Place <sigma.war> under your Tomcat installation at `C:\Program Files\Apache Tomcat 4.0\webapps`
7. 7. Start up tomcat, and it should expand itself. Then go to http://localhost:8080/sigma/KBs.jsp to get started. That page allows you to create a knowledge base and select the KIF files that comprise it.
8. Edit `C:\Program Files\Apache Tomcat 4.0\KBs\config.txt` to have the following line read:

   ```
   <preference key="inferenceEngineDir"
   value="C:\vampire\Vampire_VSWorkspace\vampire\Release"/>
   ```

   or wherever you have placed the kif.exe file as a result of step 5 above. You may also set this parameter by clicking "Preferences" from the Sigma home page.

Note that if you want to load a new version of Sigma, you must kill and restart Tomcat as well as deleting the directory
`C:\Program Files\Apache Tomcat 4.0\webapps\sigma`

User Guide

Sigma (Pease, 2003) is an environment for creating, testing, modifying and inferencing on ontologies. In this guide, we will first run through a typical session of using major Sigma functions, and then provide a reference to the remaining functions. Note that Sigma does not contain functions for editing knowledge bases. Knowledge bases in first order logic are similar to modern programming languages in complexity and expressiveness, and the most suitable tool for editing is a powerful text editor, such as one of the open source Emacs variants, or a commercial programming language editor such as Visual SlickEdit. The color coding and formatting tools offered in such editors can be very helpful. Sigma serves the same purpose as a modern IDE, supporting structured examination, project management, and debugging.

The first screen one sees is a login screen (Figure 1)

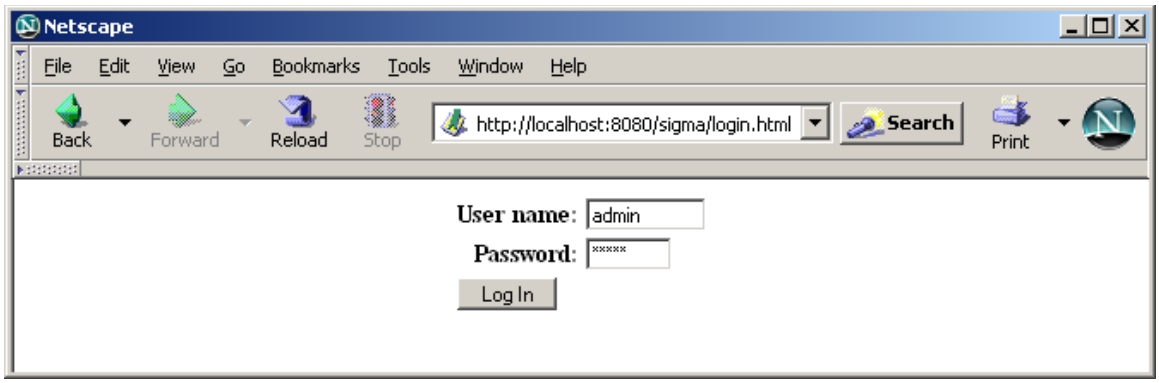

Figure 1. Login screen

In its present version, Sigma has only the most rudimentary login functionality with one hard-wired password that allows access to all Sigma functions. If that password is not chosen, only read-only operations are allowed.

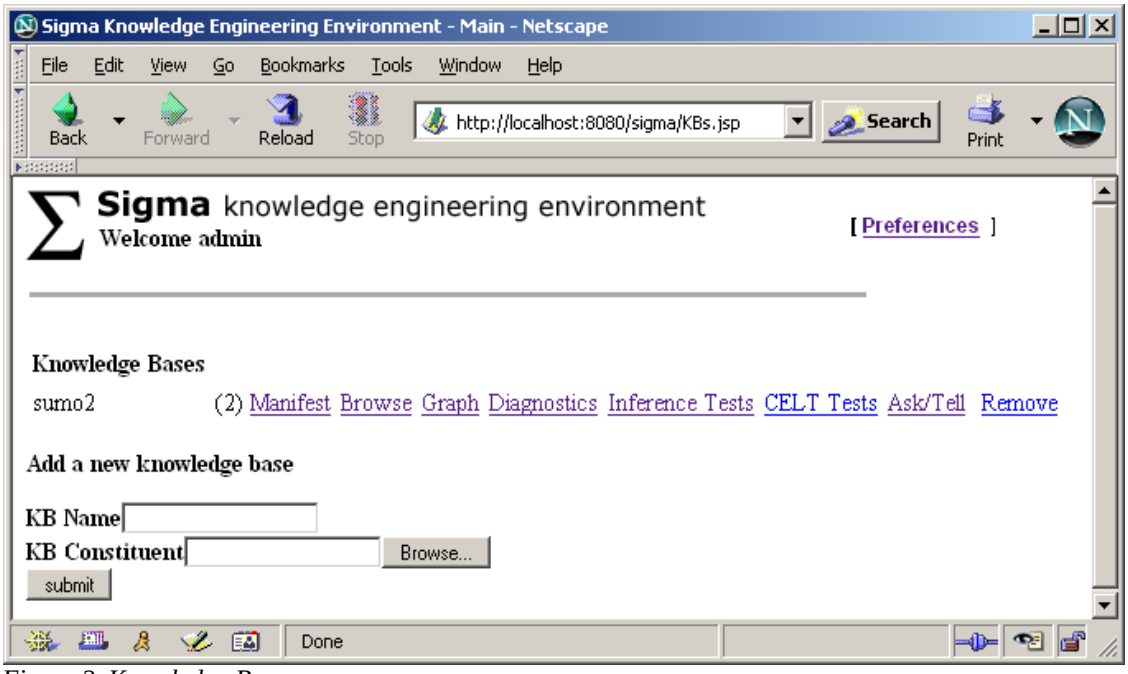

Figure 2. Knowledge Bases screen

The next screen (Figure 2) is a listing of all the knowledge bases loaded, the operations allowed on those knowledge bases, and a form for creating a new knowledge base. The standard installation of Sigma comes with the Suggested Upper Merged Ontology (Niles & Pease, 2001) pre-loaded, so you should see a screen similar to that in Figure 3.

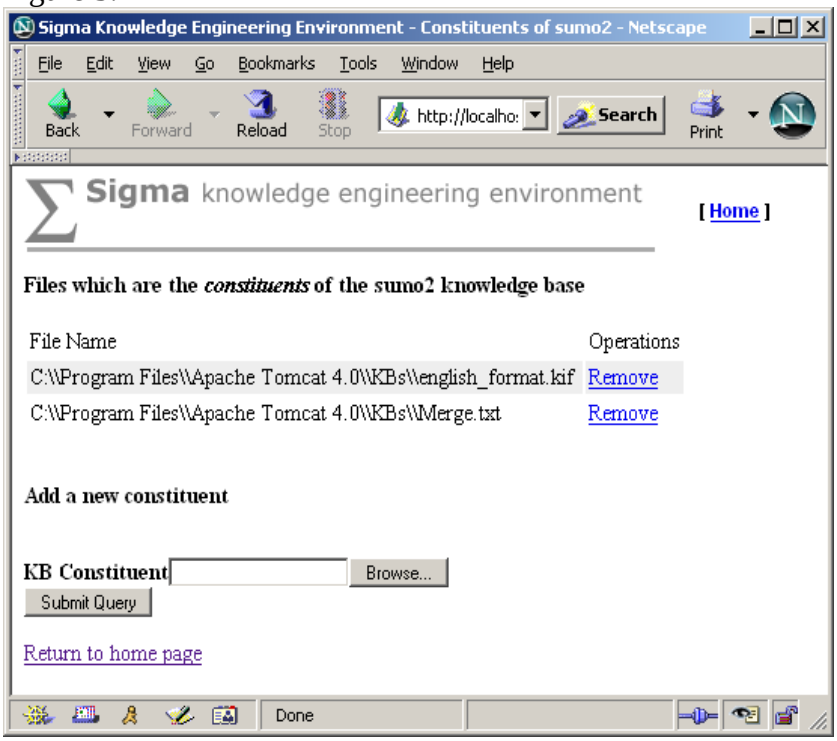

Figure 3. Manifest

Sigma organizes ontologies in *knowledge bases* which are collections of files selected by the user. Each knowledge base consists of several *constituents* which are files in Knowledge Interchange Format (KIF) (Genesereth, 1991). A particular simplified dialect of KIF is used (Pease, 2004). Each knowledge base has a *manifest* which shows the files it contains.

The most fundamental component of Sigma is a statement browser, which presents logical statements in which a given term appears. Clicking "Browse" on the Knowledge Bases screen brings you to a browser page for the selected knowledge base. The initial page just lists some statistics for the knowledge base, and provides space to type in a term from the knowledge base

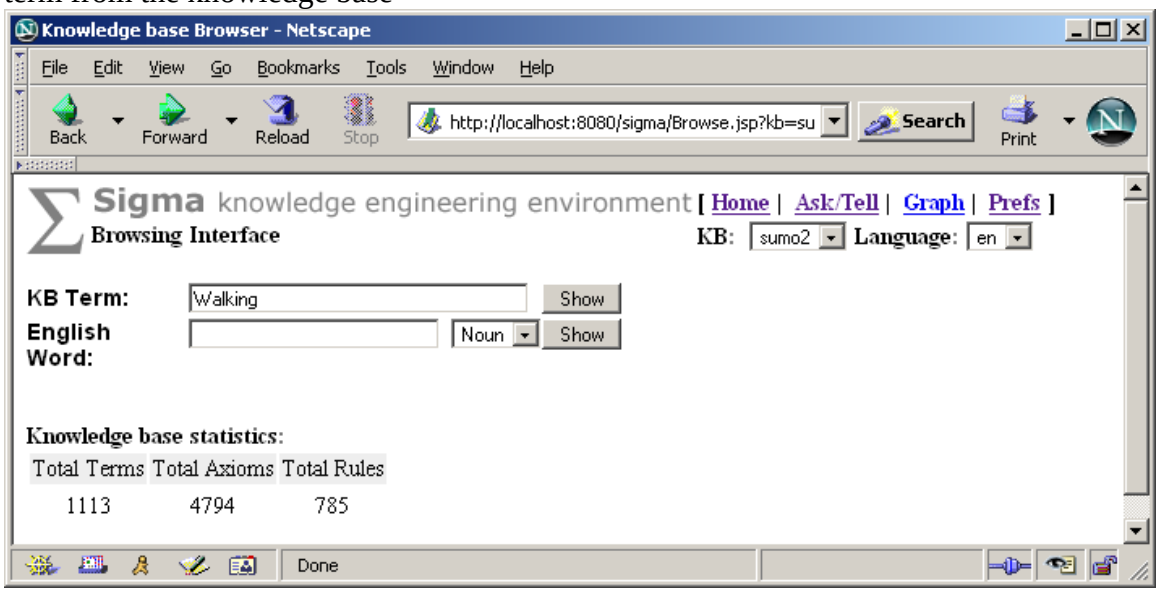

Figure 4. Initial term browser screen

In the browser screen in Figure 4, the user has typed the term Walking. The browser screen in Figure 5 shows all the statements in the knowledge base in which the term Walking appears. Clicking on a hyperlinked term in a statement brings the user to the browser page for that statement. For example, clicking on the term Running causes the browser to show all statements in which the term Running appears. The browser also shows, in the blue-grey center column, which KIF file the statement comes from, and at what line number it appears on the file. In the right hand column, an English paraphrase is shown. The paraphrases are generated automatically based on a set of format statements. On the Manifest page about, we can se the file engligh_format.kif has been loaded. That file contains language paraphrasing statements for English. In the term browser screens, one can see at the top right a menu for "Language:". That menu is constructed automatically based on the presence of language formatting statements in the knowledge base. Format files are available for SUMO in Italian, German and Hindi (among others).

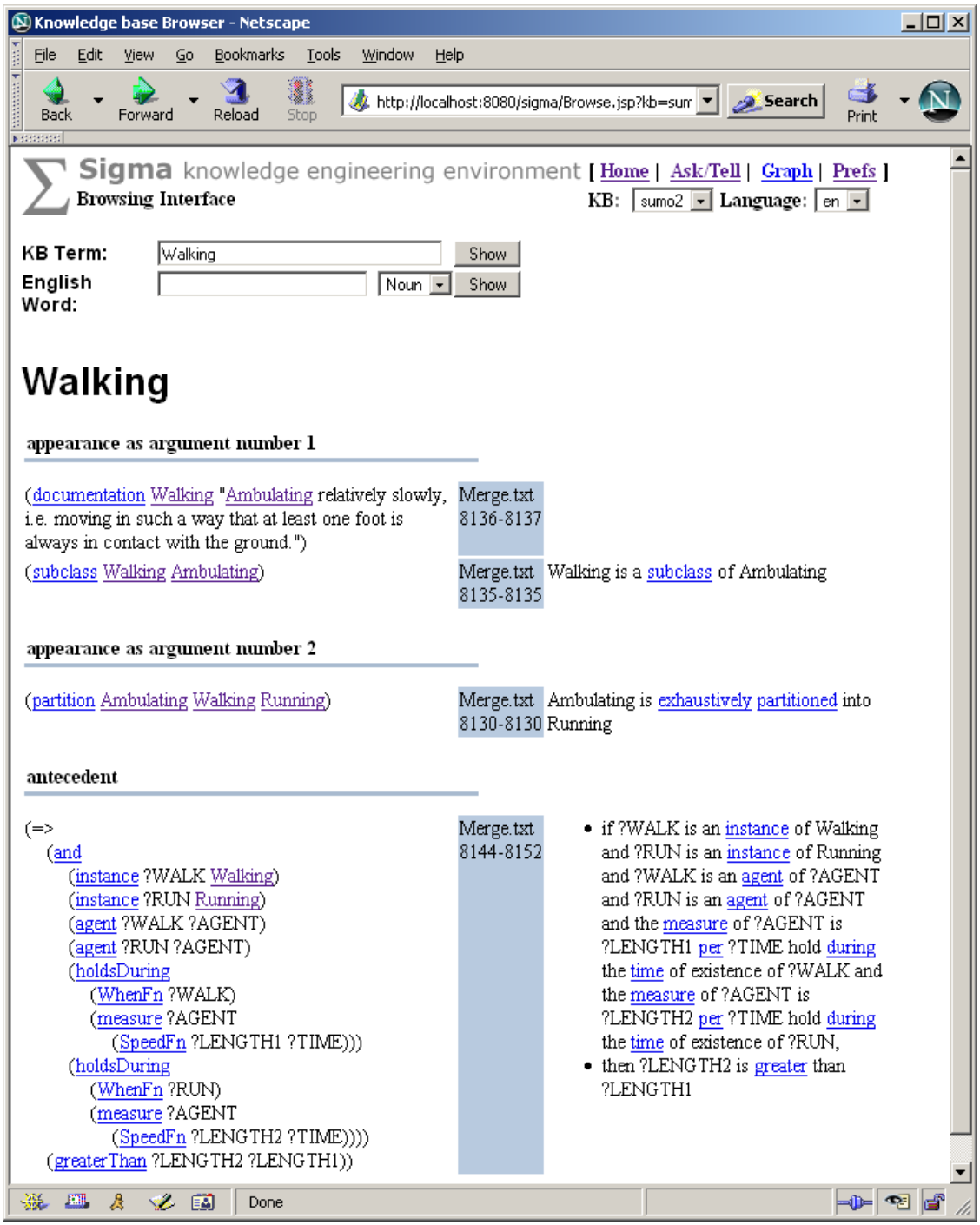

Figure 5. Term browser page for the SUMO term "Walking"

If the user types a word into the KB term box that is not found, Sigma responds with a set of terms which are closest, alphabetically, to the given term, as shown in Figure 6.

Figure 6. Term neighbors

By typing a word in the "English Word:" box, one can get a list of all English word senses in the WordNet lexicon that match, and their mappings to terms in SUMO (Niles & Pease, 2003). For example, entering the word buffalo results in Figure 7

Note that the first time the English Term function is used that there will be a delay as all the entire WordNet lexicon and SUMO mappings are loaded.

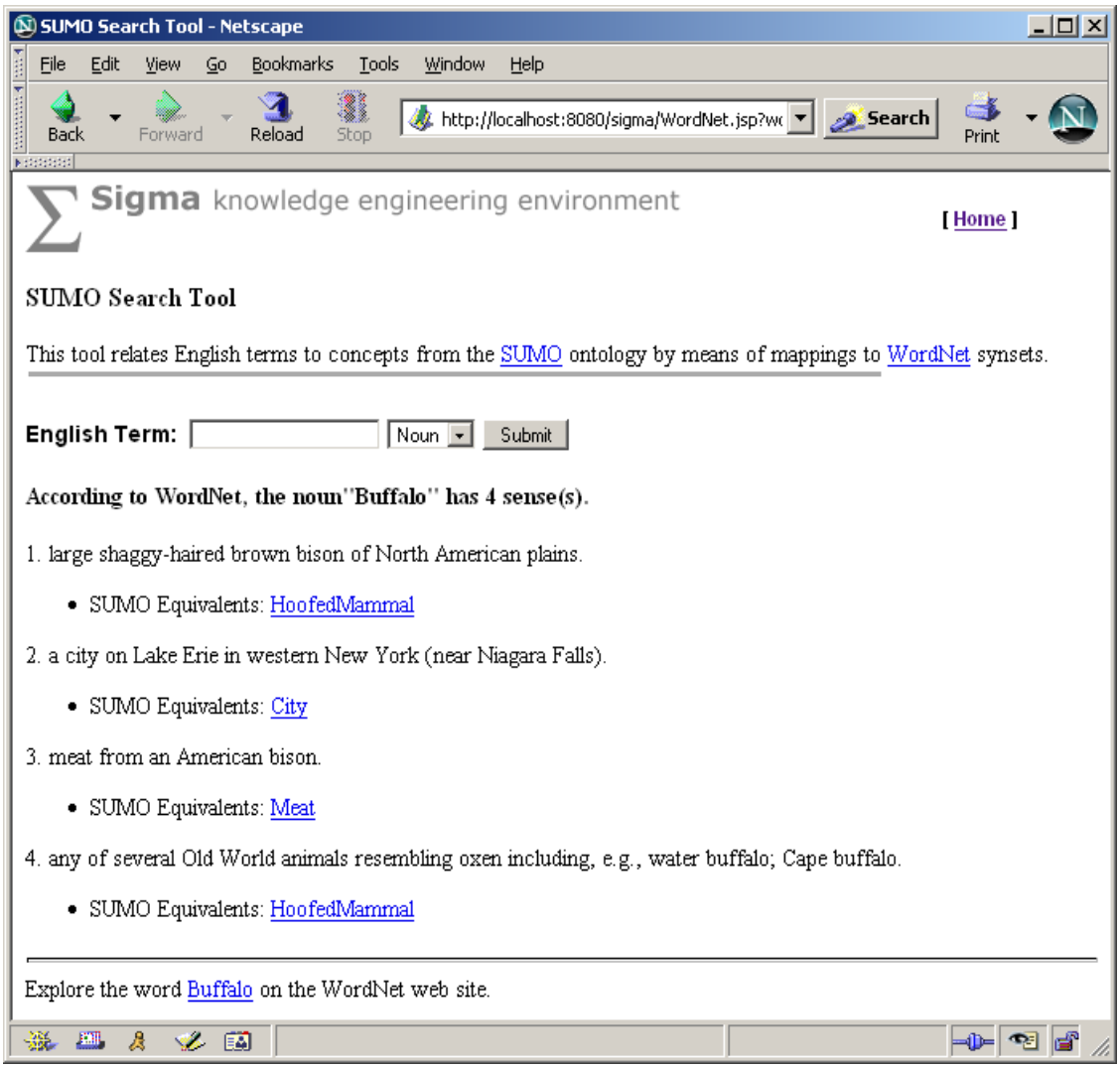

Figure 7. English word listing

Another important functionality in Sigma is for performing logical inference. From the Knowledge Bases screen, click on the Ask/Tell link. For a trivial inference, we can ask for a subclass of the class Entity. We get back a very simple proof as shown in Figure 10, that the class Abstract is a subclass of Entity. Sigma is integrated with the Vampire (Riazanov & Voronkov, 2002) theorem prover.

While a full discussion of the proofs resulting from a resolution theorem prover are beyond the scope of this manual, we can point out a few items from the proof. Each step in the proof is numbered, and each step also has a justification for how it was derived. This is either a list of numbers, for example, line 4 shows that it was derived from step 3. Currently, no further justification about the inference rule applied is provided, although in this case, one can see that 4 is derived from 3 simply by renaming the variable. Steps can also be taken directly from the knowledge base, which is denoted by "[KB]", or from the query itself. The proof method employed is proof by contradiction, where the query is negated, and the system tries to find a contradiction that results. So, we see the label

"[Negated Query]".

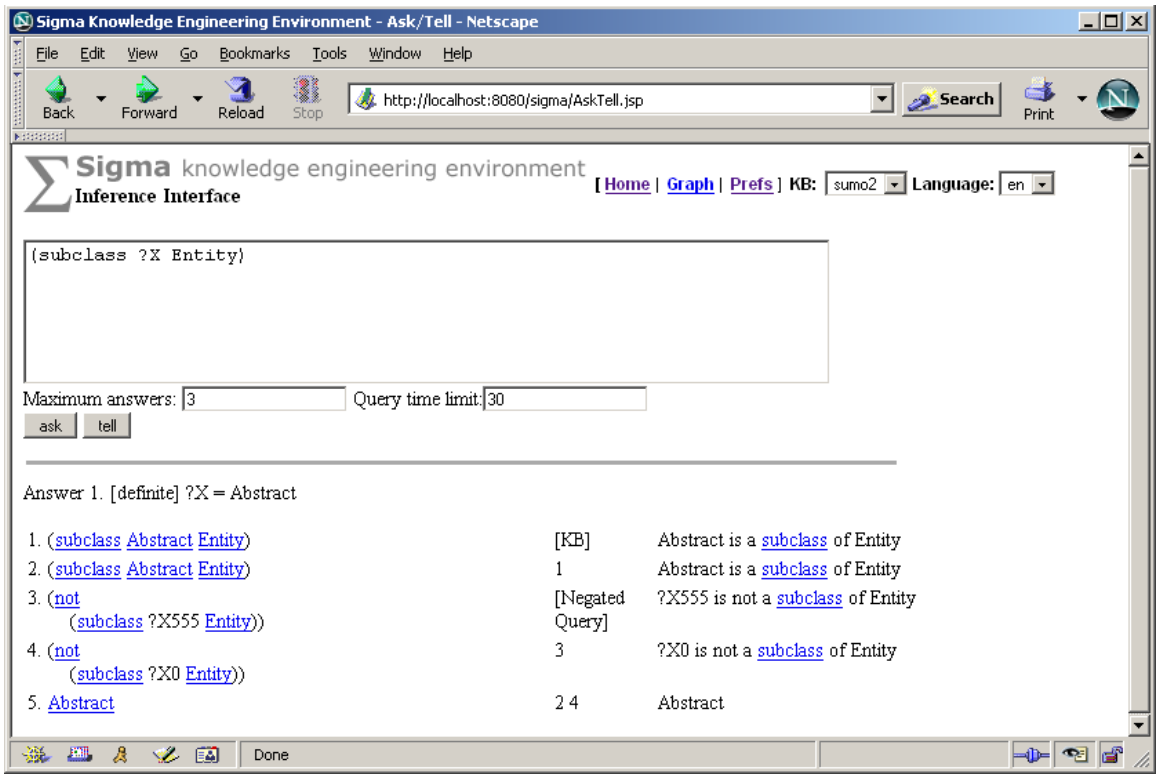

Figure 8. Sigma proof

The inference engine can be controlled by limiting the number of answers it is directed to find, as well as by providing a time cutoff. On a large, interconnected knowledge base there are so many possible search paths that the inference engine would frequently continue to search indefinitely if such cutoffs are not provided.

One can also assert an individual formula to the knowledge base by entering a KIF statement, and clicking the "tell" button. That new formula then becomes accessble for inference and browsing. If the user has not previously done a "tell" to the current knowledge base, the system created a file called "_userAssertions.kif", and add the formula, as well as any subsequent formulas, to that file. The file will be loaded automatically when Sigma is restarted, and it can also be deleted from the manifast like any constituent if desired.

Note that first order inference is computationally expensive and expected results may not be achieved, even if they logically follow from the knowledge base. Also, Vampire has no notion of what it means to return a common-sense answer, just a logically correct one, so general axioms in SUMO can occasionally give rise to answers which although logically true, may not be useful or expected. A final caution on inference is that although SUMO assumes a sorted logic, Vampire is currently unsorted, and therefore axioms can occasionally be employed in ways that are inconsistent with the argument types defined for SUMO relations, resulting in spurious answers. We hope to remedy this in a future version.

Sigma has a simple caching function that may improve many inferences, since reasoning about subclass relationships often necessary. Sigma can compute the transitive closure of subclasses statements and assert them directly, so that Vampire does not need to apply the subclass reasoning axiom in SUMO during inference, but can find the ground statement directly. For example, if we ask whether Human is a subclass of Object, Vampire will have to apply the same axioms several times. It may in fact find another search path and get stuck, since it doesn't know what common-sense answer we are looking for. By asserting directly that Human is a subclass of Object, we can short-circut a number of spurious inference paths, and get better and faster results. Caching is turned on from the Preferences "[Prefs}" page.

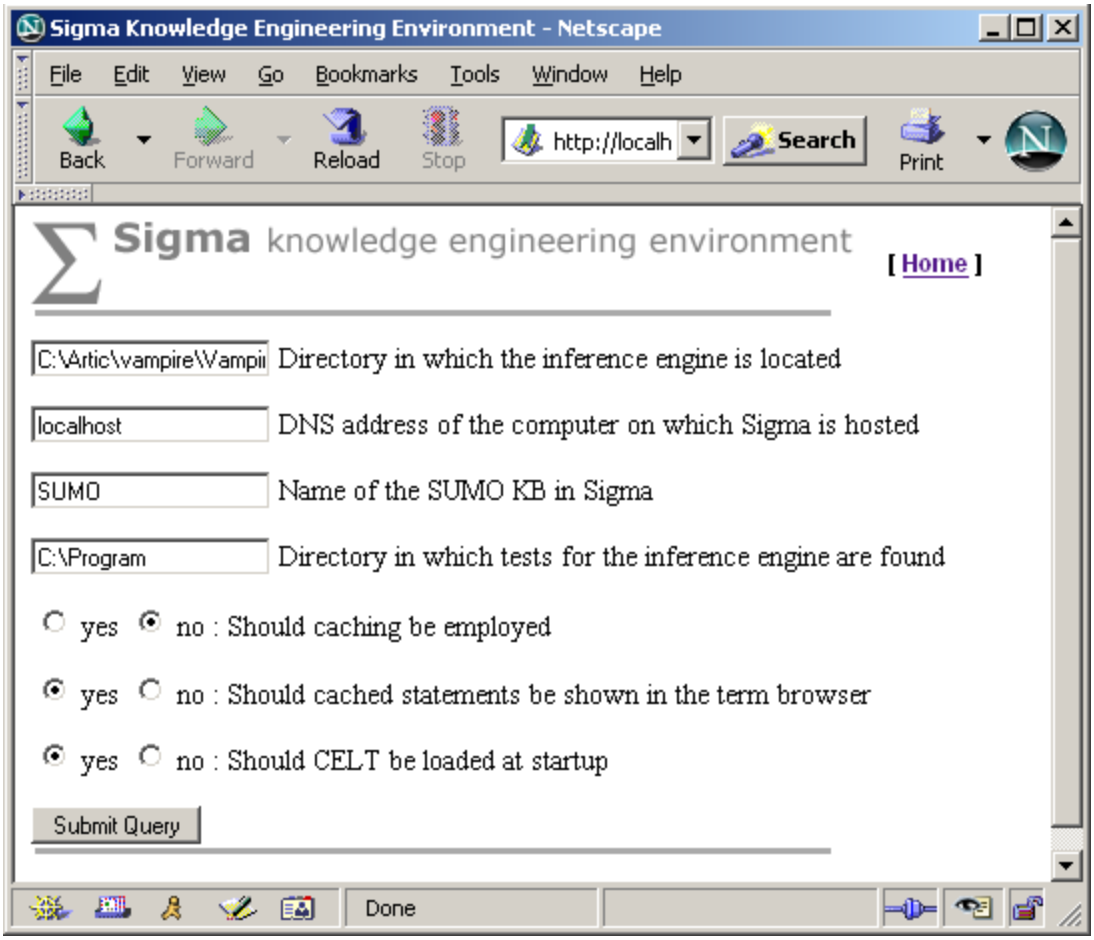

Figure 9. Preferences page

This page also allows the user to set the directory in which the inference engine executable is found, the address of the server which is running Sigma, the name of the KB containing SUMO, directory in which inference tests are found, whether automatically cached statements should be displayed in the browser, and whether the Controlled English to Logic Translation (Pease & Murray, 2003) system should be loaded at startup. This component is experimental, and not generally included in the Sigma distribution.

Reference Guide

The reference guide is organized according to the web/jsp pages available to the user in Sigma.

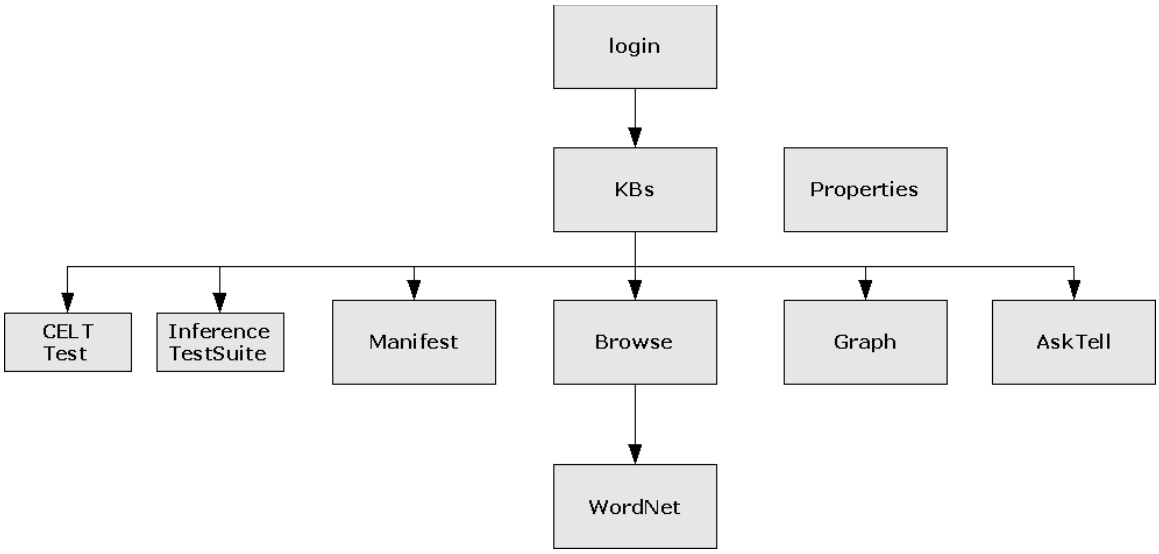

Figure 10. Pages and navigation in Sigma

login
This page allows the user to type in a login and password. In the current version of Sigma, an administrative password grants full access to Sigma functions, and all other user name and password combinations result in the availability of strictly read-only functions.

KBs
This page lists all the knowledge bases loaded into Sigma. Each knowledge base can be examined or manipulated through a set of functions, which are

- *Manifest* – takes the user to the Manifest page, which shows all the files which comprise the knowledge base
- *Browse* – takes the user to the Browse page, where all the statements for a given term are presented, and the user may also search for the alphabetical neighbors of a term, and for English words and their links to SUMO.
- *Graph* – takes the user to the Graph page which presents a graph of terms for a given relation
- *Diagnostics* – takes the user to the Diag page which provides diagnostics on the knowledge base
- *Inference Tests* – takes the user to the InferenceTestSuite page where files in a selected directory, which specify inference tests, are executed and tallied
- *CELT Tests* - takes the user to the InferenceTestSuite page where files in a selected directory, which specify CELT tests, are executed and tallied
- *Ask/Tell* – takes the user to the Ask/Tell page where one can make KIF or restricted

English statements and questions to the knowledge base
- *Remove* – delete the given knowledge base

WordNet

This page shows the WordNet synsets for the word and part of speech that he or she has selected. Each synset shows the WordNet definition, and the SUMO term or terms that are linked to that synset. Clicking on a hyperlinked SUMO term takes the user to the term browser page for that term. The user can also type in another word and select a different part of speech if desired. Sigma performs some simple processing of words that allows a word to be found even if the user types a plural or past tense, since WordNet stores only the root grammatical forms of words.

Browse

The browse page is where the ontologist is likely to spend the most time. When no term is selected, three metrics are displayed. "Total terms" are the number of names defined in the knowledge base. They may be classes or instances. Note that variables do not count. "Total Axioms" are the number of statements in the knowledge base. Note that these are user-authored statements and this measure is arguably imprecise, since different users could alternately code (=> A (and B C)) as one statement and (=> A B) (=> B C) as two statements, even though they are logically equivalent. "Total Rules" is also somewhat of an approximation. Sigma counts occurances of "=>" and "<=>". Since (=> A B) is equivalent to (or (not A) B), logically equivalent knowledge bases could have different metrics. Note also that the number of rules is a subset of the number of axioms.

The browse page has a number of controls. "KB term" allows the user to search for a term in the knowledge base, and display all statements which include that term. If the term is not found, the terms which are closest, alphabetically, are displayed. When statements are displayed, they are hyperlinked, so that when the user clicks on a term, he is automatically taken to the term browser page for the term.

The "English Word" area functions as described in the WordNet page description in this section.

There are links on this page to the "home" knowledge base listing page, the Ask/Tell page, Graph page and Pref(erences) page.

There are two menus at the top right of the page. One menu simply selects the knowledge base. This is an alternative to selecting "Browse" from a knowledge base on the KBs page, since one can select the knowledge base of interest directly, without having to return to that page. The second menu is the "language" menu.

Sigma is capable of loading KIF files which specify natural language formatting templates. These templates allow Sigma to paraphrase logical statements in a natural language. The formatting is quite simplistic, but can give some assistence to users who are either not comfortable in logic, or not comfortable in the human language which an ontology has been written. The definition language is defined in the Appendix: Natural Language Format.

The user can load several natural language format files. The selection in the language menu tells Sigma in which language to format statements. The natural language paraphrases appear in the right hand column of the statement listing.

The center column of the browser page shows the name and line number of the file that contains the statement.

Manifest

The manifest page contains a listing of all the files that comprise the selected knowledge base. Full path names are provided. Next to each file is a link to remove the file. Below the list are controls that allow a new file to be added to the knowledge base.

Properties

This page allows the user to set the directory in which the inference engine executable is found. This can also be set directly in the config.txt file as specified in the installation instructions.

The user can set address of the server which is running Sigma. This allows hyperlinks to function properly when the user is working with a Sigma installation that is on a server, rather than on his desktop.

The user should set the name of the KB containing SUMO. This allows the hyperlinks on the WordNet page to function properly.

The user can set the directory in which inference tests are found. The format for these tests is given in the Appendix: Inference Test Format.

The user can set caching should be performed, and whether automatically cached statements should be displayed in the browser. Because caching results in the transitive closure of all subclass statements this can result in a large amount of information which, although helpful to inference engine performance, is not helpful when shown in the browser.

The user can set whether the Controlled English to Logic Translation (Pease & Murray, 2003) system should be loaded at startup. Starting this component can take several minutes, so it's often helpful to turn this off when work in CELT is not being routinely performed. Also, since this component is experimental, it is not included in the general Sigma distribution and must be turned off in that case.

Graph

This page allows the user to get a hierarchical view of terms and relations in a knowledge base. Most typically, the user selects the subclass relation in order to see a portion of a class/subclass tree. The user selects a particular term and the number of “levels” above and below that term to display. Only a text view is supported at this time. The user can select a different binary relation, such as “subrelation” for example, in order to see a hierarchy of predicates. This page also includes KB and language menus as described in the reference for the Browse page. Levels in the hierarchy are indicated by degree of indenting.

AskTell

This page supports asserting new statements to the knowledge base, and posing logical queries to the knowledge base. If the CELT system has been loaded, all non-KIF statements are assumed to be natural language statements, and are passed to CELT. CELT queries are terminated with a question mark, just like an English question. All

other sentences are assumed to be statements. This page includes menus for the knowledge base, and the language in which natural language paraphrases are presented (see the description of the Browse page for more detail on these controls).

Pressing, the "tell" button results in the statement being asserted to KB_userAssertions.kif, where "KB" is the name of the current knowledge base. If there is a syntax error in the statement however, it is not asserted, and an error message will appear.

Pressing the "ask" button results in a query being posed to the inference engine.The inference engine can be controlled by limiting the number of answers it is directed to find, as well as by providing a time cutoff. On a large, interconnected knowledge base there are so many possible search paths that the inference engine would frequently continue to search indefinitely if such cutoffs are not provided.

While a full discussion of the proofs resulting from a resolution theorem prover are beyond the scope of this manual, we can point out a few items from the proof. Each step in the proof is numbered, and each step also has a justification for how it was derived. This is either a list of numbers, for example, line 4 shows that it was derived from step 3. Currently, no further justification about the inference rule applied is provided, although in this case, one can see that 4 is derived from 3 simply by renaming the variable. Steps can also be taken directly from the knowledge base, which is denoted by "[KB]", or from the query itself. The proof method employed is proof by contradiction, where the query is negated, and the system tries to find a contradiction that results. So, we see the label "[Negated Query]".

InferenceTestSuite

This page will run and show the results of inference tests in the directory indicated in the Preferences. Note that all tests will be run before results are shown. This can take a long time if there are many tests. If no time limit is given for a particular test, a default of 30 seconds is used. It will be helpful to view the Tomcat window for messages showing how the tests are progressing. When the tests are complete, a list will be shown with links to each test source, its proof, if found, whether the test was successful as compared to the expected answer, and how long the test took. A total is provided for time and correctness. Appendix: Test Formats described the format of test files. This same page also runs CELT tests, if the user clicks on the "CELT Test" link from the KBs page. CELT test results are shown in three columns with the English input first, then the expected logical form, the actual logical form, and whether the test failed or succeeded. CELT test format is described in the Appendix: Test Formats.

Diag

Three basic tests are run over the entire knowledge base when this page is selected. They are:

- *Terms without documentation* – whether each term has a 'documentation' relation defined.
- *Terms without parent* – whether each term has an 'instance' or 'subclass' relation defined for it
- *Terms without a root at Entity* – whether each term ultimately is an instance or

subclass of Entity, which is the root term in SUMO

Many other tests would be valuable, but these simple tests do indicate some common problems.

References

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.

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.

Niles, I., and Pease, A., (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, Proceedings of the IEEE International Conference on Information and Knowledge Engineering. (IKE 2003), Las Vegas, Nevada, June 23-26, 2003.

Pease, A., (2003). The Sigma Ontology Development Environment, in Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems, August 9, Acapulco, Mexico.

Pease, A., (2004). Standard Upper Ontology Knowledge Interchange Format. Language manual, unpublished.

Pease, A., and Murray, W., (2003). An English to Logic Translator for Ontology-based Knowledge Representation Languages. In Proceedings of the 2003 IEEE International Conference on Natural Language Processing and Knowledge Engineering, Beijing, China, pp 777-783.

Riazanov, A., & Voronkov, A., (2002). The Design and Implementation of Vampire, AI Communications, Volume 15. Numbers 2-3.

Acknowledgements
Many thanks to Michal Sevcenko for defining the natural language format language, and to the Air Force and ARDA for their support for Sigma development.

Appendix: Natural Language Format

The 'format' predicate associates a concept (either relation or a function with a string. The format predicate takes three arguments: the name or abbreviation of a natural language, the relation name and the format string. When there is a need to visualise certain concept in natural language, the associated string is used. The string contains a natural language description of the concept and special tags which are interpreted with the browser.

The description of these tags follows:

- **&%token** - specifies a token that will be made into a hypertext link to concept being visualized
- **%1, %2, ...** - this tag will be substituted with a natural language representation of the concept's respective argument
- **%n{text}** will be replaced either with empty string, if a predicate is being rendered as positive, or 'text' otherwise; the %n tag can be used as a shortcut for %n{not}
- **%p{text}** replaced with 'text' for positive rendering and with an empty string for negative rendering
- **%*{range}[delim]** will be replaced with a list of natural-language representation of a subset of arguments; range specifies which arguments will be included - it is a comma separated list of numbers or ranges; for example, range '1-4,6' denotes first, second, third, fourth and sixth argument; the delim parameter specifies the delimiter which will be used to separate representations of arguments; both {range} and [delim] may be ommited - range defaults to all arguments, and delim defaults to a single space
- **%%** will be replaced with a single percent character

The 'termFormat' predicate relates a term to a natural language presentation of that term. It takes three arguments, the name or abbreviation of a natural language, the term name and the format string

Appendix: Test Formats

Inference test files are legal KIF files. They must end with the extension ".tq". There are several special purpose predicates that are supported. 'note' is a unary predicate that takes a term which will be the identifier for the test. 'query' is a unary predicate that has a KIF query as its only argument. 'answer' is a unary predicate whose argument may be either 'yes', 'no' or a list pair of variable name and value. 'time' is a unary predicate whose argument specifies the number of seconds the system should wait for an answer.

There may also be a single CELT test file in the same test directory. It must be named celtTest.txt. It must be legal KIF. It contains pairs of statements. The first special unary predicate is 'sentence' which takes a string containing a CELT sentence as its argument. The second special unary predicate is 'answer' which takes a KIF formula as its argument. The output of CELT is compared against the answer formula to determine the success or failure of the test.