

Sigma Knowledge Engineering Environment Installation Instructions and User Guide

for Sigma version 1.5 beta

08/21/2006

written by Adam Pease

[apeace @ articulatesoftware.com](mailto:apeace@articulatesoftware.com)

License

This code and documentation is copyright Articulate Software (c) 2005, 2006. Some portions copyright Teknowledge (c) 2003 and reused under the terms of the GNU license. This software is released under the GNU Public License <http://www.gnu.org/copyleft/gpl.html> .

Users of this code also consent, by use of this code, to credit Articulate Software and Teknowledge in any writings, briefings, publications, presentations, or other representations of any software which incorporates, builds on, or uses this code. Please cite the following article in any publication with references:

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.

Note that the current release of Sigma is "beta" code, and not suitable for any mission-critical use. It is strictly a tool for research. Sigma comes with no warranty, expressed or implied. By using Sigma, you assume all responsibility for events it may result in, and agree to hold Articulate Software blameless for any adverse consequences.

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

Table of Contents

Sigma Knowledge Engineering Environment.....	1
Installation Instructions and User Guide.....	1
1 Introduction.....	4
2 Sigma Installation.....	4
2.1 Quick Start.....	4
2.2 Prerequisites.....	7
2.3 Additional Installation Notes.....	7
2.3.1 On a Windows Host.....	8
2.3.2 On a Linux Host.....	9
2.3.3 Instructions Pertinent to All Platforms.....	11
2.4 Troubleshooting.....	12
3 User Guide.....	12
4 Reference Guide.....	22
5 References.....	27
6 Acknowledgements.....	27
7 Appendix: Natural Language Format	27
8 Appendix: Test Formats.....	28

1 Introduction

Sigma (Pease, 2003) is an environment for creating, testing, modifying, and performing inference with ontologies. This document explains how to install and use Sigma. Those new to Sigma are advised to read the entire document, but experienced users who simply want to install a newer version of Sigma might be able to get by with reading just Section 2.1 Quick Start. Please bear in mind, however, that Sigma is a work in progress with a few rough edges, and that one purpose of this document is to help you achieve productive use of Sigma. Taking time to read it now might save you time in the future.

2 Sigma Installation

Section 2.1 is a “quick start” guide intended for users who wish to proceed with installation before reading a more comprehensive overview of Sigma’s components and functionality. Note that the quick install will not work in all cases, and users who encounter problem will have to read Section 2.3. Section 2.2 describes Sigma’s principal components, indicates where they can be obtained, and notes some of the assumptions that apply to Sigma’s installation and use. Section 2.3 provides additional notes on installation for those who cannot use, or prefer not to use, the `InstallSigma` script, and explains aspects of the implementation that might help you to solve any problems you encounter. **Please note that there is no Linux install script at this time.**

2.1 Quick Start

1. Install version 5.0 or more recent of a Java Runtime Environment (JRE). You can obtain a JRE at <http://java.sun.com>.
2. Install version 5.5 or more recent of Tomcat. Tomcat is available at: <http://jakarta.apache.org/tomcat/>. Note that hereafter, we will refer to the directory in which Tomcat is installed as [Tomcat]. The default is `/var/tomcat5` on Linux and `C:\Program Files\Apache Software Foundation\Tomcat 5.5` on Windows.
3. Make sure Tomcat is properly configured for Sigma. In Windows, this can be accomplished by using the Configure Tomcat utility accessible through the Start menu:

```
Start
->All Programs
    ->Apache Tomcat 5.x
        ->Configure Tomcat
            ->Startup: Working Path=[Tomcat]

Start
->All Programs
    ->Apache Tomcat 5.x
        ->Configure Tomcat
            ->Java: Set Initial and Max Memory Pool to
300MB
```

In Linux, modify the startup script for Tomcat to use a large stack size. Edit [Tomcat]/bin/catalina.sh to add the command

```
CATALINA_OPTS="$CATALINA_OPTS -Xmx300M"
```

For both Windows and Linux, make sure environment variables such as JAVA_HOME have been properly set to support Tomcat.

4. Install WordNet 2.1 or a more recent version. WordNet is available at <http://wordnet.princeton.edu/obtain>.
5. Download and unpack the Sigma install package, sigma-latest.zip, available at this URL: http://sourceforge.net/project/showfiles.php?group_id=102489. The install package will create a directory named sigma-latest in the directory where it is unpacked.
6. Open the directory sigma-latest. It should contain several files and subdirectories, approximately like those shown in Figure 1.

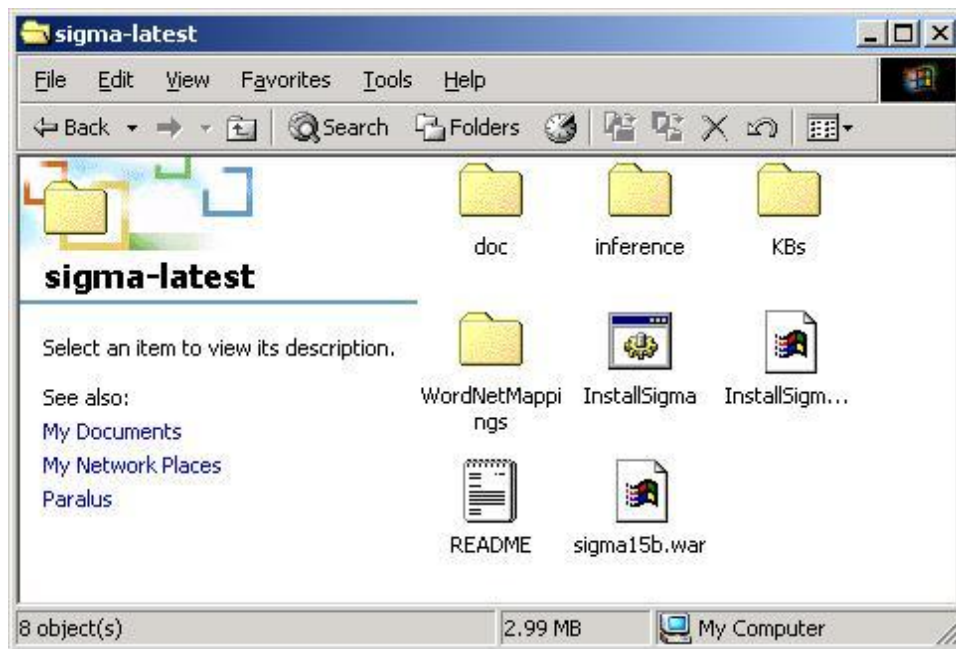


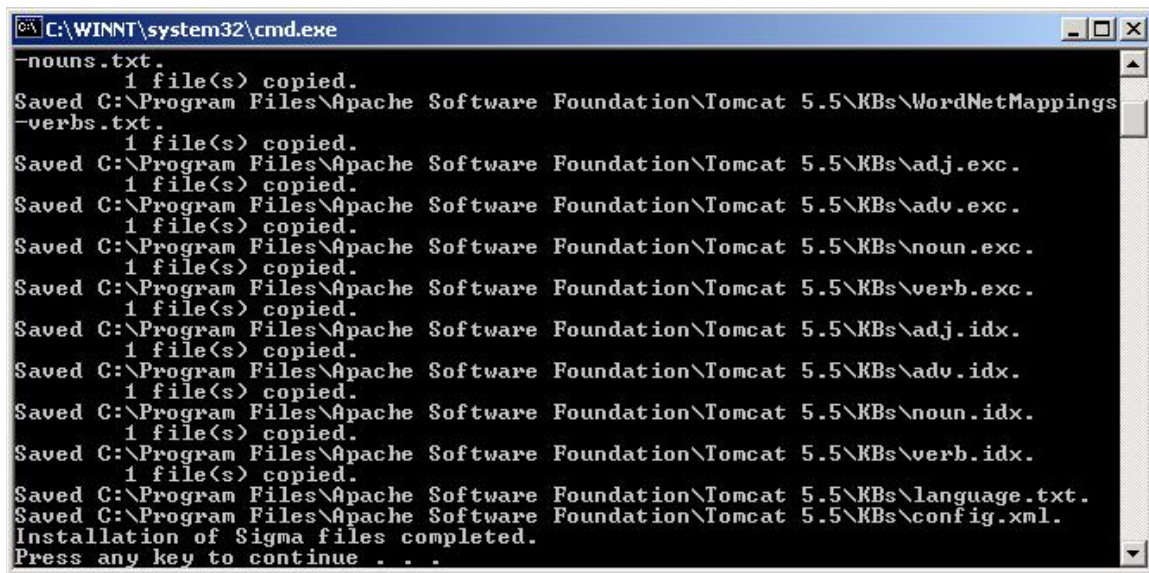
Figure 1: The Sigma release package directory

7. Make sure the Tomcat server is not running. You can stop it with the Tomcat Monitor application.

```
Start
->All Programs
   ->Apache Tomcat 5.x
       ->Configure Tomcat
           ->General
               ->Stop
```

8. Run the relevant, platform-specific version of the script named InstallSigma, which is located in sigma-latest. On a Windows host, you will run the file named InstallSigma.bat, either by invoking it from a command prompt window or by

double-clicking on the file icon with your mouse pointer. The file icon will have a small gear in the center, as depicted in Figure 1. On a Linux host, you will run the shell script named `InstallSigma.sh` by invoking it from the command line prompt in a terminal window. `InstallSigma`'s main purpose is to copy various files from the `sigma-latest` install package and the WordNet installation into the directories under [Tomcat] named `KBs` and `webapps`. If `InstallSigma` encounters a problem that prevents it from copying a file or completing some other action, it will print a message noting the failure and will then quit, giving you a chance to diagnose and fix the problem. If `InstallSigma` successfully finishes its programmed tasks, it will print the message, `Installation of Sigma files completed.` (Figure 2)



```
C:\WINNT\system32\cmd.exe
-nouns.txt.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\WordNetMappings
-verbs.txt.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\adj.exc.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\adv.exc.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\noun.exc.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\verb.exc.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\adj.idx.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\adv.idx.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\noun.idx.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\verb.idx.
  1 file(s) copied.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\language.txt.
Saved C:\Program Files\Apache Software Foundation\Tomcat 5.5\KBs\config.xml.
Installation of Sigma files completed.
Press any key to continue . . .
```

Figure 2: Install script output

9. After successfully running `InstallSigma`, restart Tomcat, and then direct your web browser to this URL: <http://localhost:8080/sigma/KBs.jsp> . You may experience a brief delay as Tomcat unpacks and processes Sigma's web archive, but after a few seconds the Sigma login page should appear (Figure 3).



Figure 3: Login screen

10. You should be able to log in with the username `admin` and the password `admin`.

See Section 2 of this document, User Guide, for instructions on creating, modifying, and working with knowledge bases. Note that you will probably have to modify Sigma's default configuration settings to enable complete functionality, since some of the values are not set during the installation procedure, and others are based only on educated guesses. Follow the [Preferences] link to update your configuration settings. To save your configuration settings, be sure to click on the button anomalously named "Submit Query".

2.2 Prerequisites

Sigma consists of, and depends on, several components and resources:

1. A host (computer) with at least 512 MB of main memory, running either a Linux or Windows (NT, 2000, XP, 2003) operating system;
2. A web browser, such as Internet Explorer or Firefox, because Sigma's primary user interface consists of dynamically generated web pages;
3. Version 5.0 or more recent of a Java Runtime Environment (JRE), which makes it possible for compiled Java programs to run on the host. This can be obtained from the relevant download area at <http://java.sun.com> .
4. Version 5.5 or more recent of the Jakarta Tomcat web application server, which is a Java program that runs on the host and acts as a convenient "container" and communication protocol (HTTPD) manager for the smaller Java programs that implement many of Sigma's capabilities. This can be obtained from the relevant download area at <http://jakarta.apache.org/tomcat/> .
5. WordNet is an English lexicon and semantic network. It can be obtained at <http://wordnet.princeton.edu/obtain> . The relevant files are contained in the directory named `dict`, under [WordNet] (the WordNet base install directory). At the time of writing, version 2.1 is the most recent version of WordNet.

These components constitute the basic computing and software infrastructure on which Sigma depends, and are NOT included in the Sigma release package.

2.3 Additional Installation Notes

The instructions in this section expand upon and, to some extent, recapitulate the instructions provided in Sections 1.1 and 1.2, above. They are intended especially for users who might not want to use the script `InstallSigma` or for whom `InstallSigma` will not work, due, perhaps, to a complex or non-standard host configuration.

These installation instructions assume that if the target host is running Windows, Tomcat and WordNet will be installed in the default locations (*i.e.*, under `C:\Program Files\Apache Software Foundation\Tomcat 5.x\` and `C:\Program Files\WordNet\2.1\`, respectively). Please note that in this document, the base install location (root directory) of Tomcat will be referred to as [Tomcat], and the base install location of WordNet will be referred to as [WordNet].

2.3.1 On a Windows Host

1. If necessary, download and install a Java Runtime Environment (JRE), version 5.0 or later, available at <http://java.sun.com> .
2. Download and install the Tomcat web application server, version 5.5 or later, available at <http://jakarta.apache.org/tomcat/> . Create a directory named KBs under [Tomcat] (*i.e.*, under the base Tomcat install directory).
3. Download and unpack the Sigma install package, `sigma-latest.zip`, available at this URL: http://sourceforge.net/project/showfiles.php?group_id=102489 . The install package will create a directory named `sigma-latest` in the directory where it is unpacked.
4. If you intend to use the default SUMO KB, copy the files `Merge.kif` and `english_format.kif` from `sigma-latest\KBs` into `[Tomcat]\KBs`. If you wish to add other KIF files, or you intend to create a new SUMO KB from scratch, do not copy any KIF files into `[Tomcat]\KBs`. Instead, keep KB constituent KIF files in the install package's KBs directory, or some other directory of your choice. Sigma copies KB constituent source files from their origin directory to `[Tomcat]\KBs` as it builds a new composite KB. If the files are read from `[Tomcat]\KBs` and then copied back into `[Tomcat]\KBs`, they will be corrupted in the process. In other words, creating a KB from scratch requires that the KIF KB constituent files NOT be in the `[Tomcat]\KBs` directory at the outset, but rather be stored in some other directory (such as `sigma-latest\KBs`).
5. The Vampire inference engine executable is contained in the install package as `sigma-latest\inference\kif.exe`. It can also be downloaded separately as `kif.exe` from the Sigma download area on SourceForge: <http://prdownloads.sourceforge.net/sigma/kee/kif.exe?download> . If you want to compile Vampire from source code, the code and Makefile are available in Sigma's CVS repository: <http://sigmakee.cvs.sourceforge.net/sigmakee/Vampire/> .
6. Copy the Sigma web application archive, `sigma-latest\sigma[xxx].war`, into `[Tomcat]\webapps`. The most recent version of the archive can also be obtained from Sigma's SourceForge download site, *e.g.*, <http://prdownloads.sourceforge.net/sigma/kee/sigma15b.war?download> . Note that you should rename the archive file to `sigma.war` when you copy it into `[Tomcat]\webapps`, *e.g.*, `sigma15b.war => [Tomcat]\webapps\sigma.war`.
7. Configure Tomcat by using the Configure Tomcat utility accessible through the Start menu:

```
Start
->All Programs
    ->Apache Tomcat 5.x
        ->Configure Tomcat
            ->Startup: Working Path=[Tomcat]
```

```
Start
->All Programs
    ->Apache Tomcat 5.x
        ->Configure Tomcat
```


->Java: Set Initial and Max Memory Pool to
300MB

Also, be sure to properly set any environment variables that might be required to support Tomcat, such as JAVA_HOME.

8. Start up Tomcat, *e.g.*, by using the Monitor Tomcat utility. The Sigma web application archive, [Tomcat]\webapps\sigma.war, should be automatically expanded.
9. Direct your browser to <http://localhost:8080/sigma/KBs.jsp> to get started. After logging in, you will be able to select KIF KB constituent files and create a knowledge base comprising them. Detailed instructions on creating a knowledge base follow in Section 2, User Guide.
10. Check the file [Tomcat]\KBs\config.xml to make sure the value for the key “inferenceEngineDir” is properly set:

```
<preference key="inferenceEngineDir"
            value="C:\sigma-latest\inference\"/>
```

The value should correspond to the absolute path of the directory in which the Vampire inference engine executable is located. If the file is missing, you can set this and other configuration parameters by following the [Preferences] link from the Sigma home page. If you expect to use the default SUMO KB included in the sigma-latest.zip install package, you should make sure that the value for the key “sumokbname” corresponds to the part of the KB archive file name that precedes “-v.kif”. If the filename were SUMO-v.kif, for example, then the value for the key “sumokbname” should be “SUMO”.

2.3.2 On a Linux Host

1. If necessary, download and install a Java Runtime Environment (JRE), version 5.0 or later, available at <http://java.sun.com> .
2. Download and install the Tomcat web application server, version 5.5 or later, available at <http://jakarta.apache.org/tomcat/> . Create a directory named KBs under tomcat.
3. If the Linux install package seems not to have the contents of the tomcat/bin directory, get the missing files from the Windows version (see Section 1.3.1, above). Specifically, you will want to copy the *.sh files into tomcat/bin. You will also need to copy the files from [Tomcat]\webapps\examples of the Windows installation to the corresponding Linux directory if you want to run the standard Tomcat examples, or want to have Tomcat respond with a verification screen when you visit <http://localhost:8080> .
4. Configure environment variables for Tomcat:
 - a) Set JAVA_HOME in your .bashrc file to point to the JRE’s location.
 - b) Set CATALINA_HOME in your .bashrc file to point to [Tomcat]

- c) Add `$JAVA_HOME/bin` to your `PATH` variable.
5. Modify the startup script for Tomcat to use a large stack size. Edit `[Tomcat]/bin/catalina.sh` to add the command
 - a) `CATALINA_OPTS="$CATALINA_OPTS -Xmx300M"`
6. Download and unpack the Sigma install package, `sigma-latest.zip`, available at this URL: http://sourceforge.net/project/showfiles.php?group_id=102489 . The install package will create a directory named `sigma-latest` in the directory where it is unpacked.
7. If you intend to use the default SUMO KB, copy the files `Merge.kif` and `english_format.kif` from `sigma-latest/KBs` into `[Tomcat]/KBs`. However, if you intend to create a new SUMO KB from scratch, do not copy any KIF files into `[Tomcat]/KBs`. Instead, keep KB constituent KIF files in the install package's KBs directory, or some other directory of your choice. Sigma copies KB constituent source files from their origin directory to `[Tomcat]/KBs` as it builds a new composite KB. If the files are read from `[Tomcat]/KBs` and then copied back into `[Tomcat]/KBs`, they will be corrupted in the process. In other words, creating a KB from scratch requires that the KIF KB constituent files NOT be in the `[Tomcat]/KBs` directory at the outset, but rather be stored in some other directory (such as `sigma-latest/KBs`).
8. You may be able to use the Vampire Linux executable included in the install package, `sigma-latest/inference/kif-linux`. This executable file is also available for download on SourceForge at <http://prdownloads.sourceforge.net/sigma/kee/kif-linux?download>. Note that the executable file should be renamed to `kif` after it has been downloaded. If you want to compile Vampire from source code, the code and Makefile are available in Sigma's CVS repository: <http://sigmakee.cvs.sourceforge.net/sigmakee/Vampire/>. Compile Vampire by executing `make` after copying the `/Vampire` directory to your hard drive. You must use `gcc` version 2.95.3 to compile.
9. Copy the Sigma web application archive, `sigma-latest/sigma[xxx].war`, to `[Tomcat]/webapps/sigma.war`. The most recent version of the archive can also be obtained from Sigma's SourceForge download site, e.g., <http://prdownloads.sourceforge.net/sigma/15b.war?download> . Note that you should rename the archive file to `sigma.war` when you copy it into `[Tomcat]/webapps`.
11. Start Tomcat by executing `[Tomcat]/bin/startup.sh`. The Sigma web archive should be automatically expanded.
12. Note that, unlike in Windows, a new window may not be spawned. In order to see Sigma's status messages, you can execute
 - `[Tomcat]/bin/catalina.sh run`instead.

13. When Tomcat is running, direct your browser to <http://localhost:8080/sigma/KBs.jsp> to get started. After logging in, you will be able to select KIF KB constituent files and create a knowledge base comprising them. Detailed instructions on creating a knowledge base follow in Section 2, User Guide.
14. Check the file [Tomcat]/KBs/config.xml to make sure the value for the key “inferenceEngineDir” is properly set:

```
<preference key="inferenceEngineDir"
            value="C:\sigma-latest\inference\"/>
```

The value should correspond to the absolute path of the directory in which the Vampire inference engine executable is located. If the file is missing, you can set this and other configuration parameters by following the [Preferences] link from the Sigma home page. If you expect to use the default SUMO KB included in the `sigma-latest.zip` install package, you should make sure that the value for the key “sumokbname” corresponds to the part of the KB archive file name that precedes “-v.kif”. If the filename were `SUMO-v.kif`, for example, then the value for the key “sumokbname” would be “SUMO”.

15. Note that if you want to load a new version of Sigma, you must kill Tomcat, delete the directory [Tomcat]/webapps/sigma, and then restart Tomcat.

2.3.3 Instructions Pertinent to All Platforms

To use the SUMO to WordNet mappings in Sigma:

1. Download and install WordNet 2.1, available here: <http://wordnet.princeton.edu/obtain>.
2. Copy the following files from [WordNet]\dict to [Tomcat]\KBs, renaming the files that begin with index, as indicated:

```
index.adv => adv.idx
index.adj => adj.idx
index.noun => noun.idx
index.verb => verb.idx
adv.exc
adj.exc
noun.exc
verb.exc
```

3. Copy the WordNet mapping files contained in `sigma-latest\WordNetMappings` to [Tomcat]\KBs, renaming as indicated below:

```
WordNetMappings*-adv*.txt => WordNetMappings-adv.txt
WordNetMappings*-adj*.txt => WordNetMappings-adj.txt
WordNetMappings*-nouns*.txt => WordNetMappings-nouns.txt
WordNetMappings*-verbs*.txt => WordNetMappings-verbs.txt
```

Note, for example, that `WordNetMappings21-nouns-allOntologies.txt` should be copied (renamed) to `WordNetMappings-nouns.txt`. The WordNet mapping files are also contained in a ZIP archive, available for download here:

<http://prdownloads.sourceforge.net/sigma/WordNetMappings.zip?download>.

To enable paraphrasing of SUO-KIF statements in English or another natural language:

1. Copy the file `language.txt` from the directory `sigma-latest\KBs` in the Sigma install package to `[Tomcat]\KBs`. This file, and others that enable support for the paraphrasing of statements in languages other than English, are available for download in the directory `KBs/Translations` of the Sigma's CVS repository, accessible here: <http://sigmakee.cvs.sourceforge.net/sigmakee/KBs/Translations/>.
2. For English, load the KB constituent file `english_format.kif` into the current composite KB if it has not already been loaded. `english_format.kif` is included in `sigma-latest\KBs`, and is available for download from Sigma's CVS repository at this URL: <http://sigmakee.cvs.sourceforge.net/sigmakee/KBs/>. Files containing format statements for languages other than English are available for download at <http://sigmakee.cvs.sourceforge.net/sigmakee/KBs/Translations/>.

Known resource limitations:

System resources are set to defaults suitable for most anticipated users of Sigma, but as currently configured, KIF KB constituent files larger than 1MB cannot be loaded to build a KB. Users who want to load files larger than 1MB will have to modify a line in the source file `AddConstituent.jsp`, as follows, and then recompile:

```
OLD: multiPartRequest = new MultipartRequest(request,srcDir);
```

```
NEW: multiPartRequest = new MultipartRequest(request,srcDir,5000000);
```

2.4 Troubleshooting

One common problem we are aware of is actually an issue with Tomcat itself. Pre-existing software may use some of the ports that Tomcat requires. In particular, some users have found a conflict with Tomcat's "shutdown" port of 8005. If Tomcat does not start properly, it may be necessary to edit `[Tomcat]\config\server.xml` and change this to a different port, such as 8015.

3 User Guide

In this guide, we first run through a typical session illustrating major Sigma functions, and then provide a reference to the remaining functions. Note that Sigma does not contain functions for editing knowledge bases (KBs). KBs 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 variants

of Emacs, 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 the login screen depicted in Figure 3. 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. If you do not have the administrative password to Sigma and wish to perform operations that are not read-only, log in with the user name `admin` and the password `admin`.



Figure 4: The Knowledge Bases screen

The next screen, depicted in Figure 4, displays a listing of all the composite KBs loaded, the operations allowed on those KBs, and a type-in area for creating a new composite KB. The standard Sigma release package includes the Suggested Upper Merged Ontology (Niles & Pease, 2001), so you should see a screen similar to that shown in Figure 4 if you have installed Sigma with the installation script, `InstallSigma`. If you are not using the standard release package, you will probably see a message indicating that no KBs are loaded. You will have to log in to Sigma with the administrator password and create a new knowledge base.

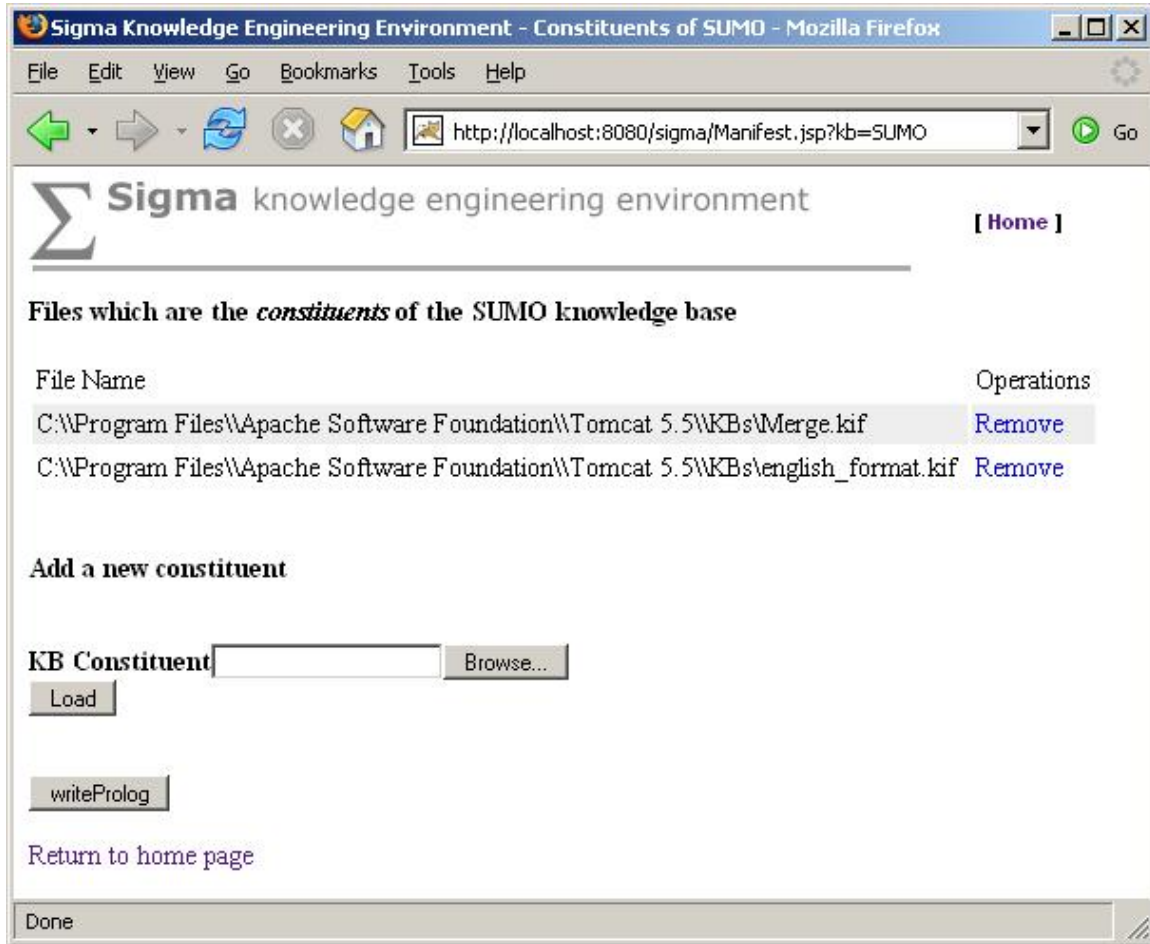


Figure 5: Manifest

Sigma organizes ontologies in *knowledge bases* that are collections of files selected by the user. Each knowledge base consists of one or more *constituents*, which are files of statements written in the 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 (Figure 5).

The fundamental interface component of Sigma is a statement browser that displays the logical statements in which a given term appears. Clicking “Browse” on the Knowledge Bases screen displays a browser page for the selected knowledge base. The initial page just lists some statistics for the knowledge base, and provides a type-in area for entering terms from the knowledge base. Figure 6 shows a browser screen in which the user has typed the term `Walking`. Figure 7 shows a browser screen that lists all of the statements in the knowledge base in which `Walking` appears.

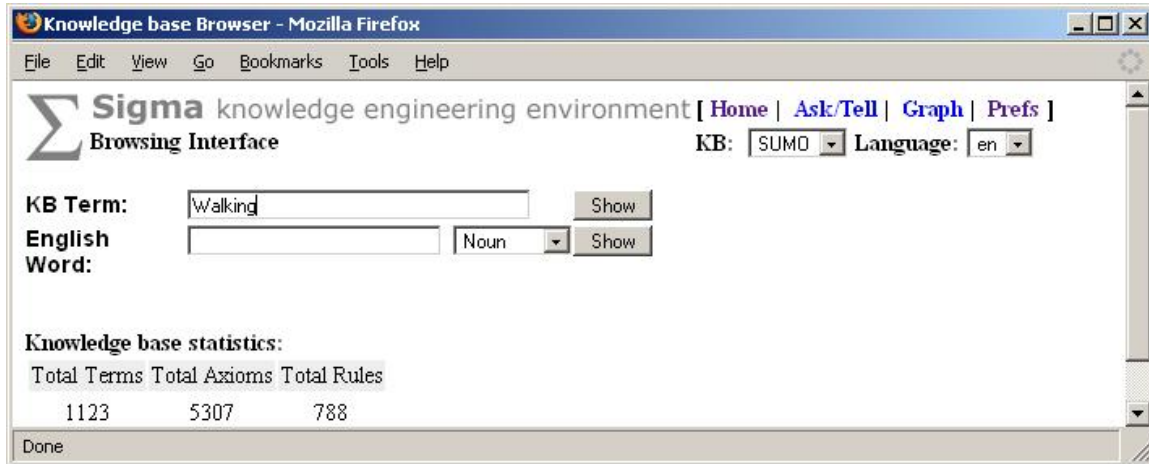


Figure 6: Initial term browser screen

The screenshot shows the 'Sigma knowledge engineering environment' interface in a Mozilla Firefox browser. The page title is 'Sigma knowledge engineering environment' with navigation links: [Home | Ask/Tell | Graph | Prefs]. Below the title is the 'Browsing Interface' section with 'KB: SUMO' and 'Language: en'. The search area shows 'KB Term: Walking' and 'English Word: Noun'. A list of related terms is displayed: *afloat, 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, en, escalate, exhibit, filter, fast_break, file, file_in, file_out, fire_walking, flounce, flounder, foot, footer, footslog, footstep, forage, gallop, gallop, gimp...*

Walking(walking)

appearance as argument number 1

(documentation Walking "Ambulating relatively slowly, i.e. moving in such a way that at least one foot is always in contact with the ground.")	Merge.kif 8177-8178	
(subclass Walking Ambulating)	Merge.kif 8176-8176	walking is a subclass of Ambulating

appearance as argument number 2

(partition Ambulating Walking Running)	Merge.kif 8171-8171	Ambulating is exhaustively partitioned into Walking Running
(termFormat en Walking "walking")	english_format.kif 793-793	termFormat en walking "walking"

antecedent

(=> (and (instance ?WALK Walking) (instance ?RUN Running) (agent ?WALK ?AGENT) (agent ?RUN ?AGENT) (holdsDuring (WhenFn ?WALK) (measure ?AGENT (SpeedFn ?LENGTH1 ?TIME))))	Merge.kif 8185-8193	<ul style="list-style-type: none"> • if ?WALK is an instance of walking and ?RUN is an instance of Running and ?WALK is an agent of ?AGENT and ?RUN is an agent of ?AGENT and the measure of ?AGENT is ?LENGTH1 per ?TIME hold during the time of existence of ?WALK and the measure of ?AGENT is ?LENGTH2 per ?TIME hold during the time of existence of ?RUN,
---	------------------------	--

Figure 7: Term browser page for the SUMO term “Walking”

Clicking on a hyperlinked term in a statement displays the browser page for that statement. Clicking on the term *Running*, for example, causes the browser to show all statements in which *Running* appears. The browser also shows, in the blue-grey center column, the name of the KIF file from which the statement was loaded, and the line number at which the statement appears in the file. An English paraphrase of the

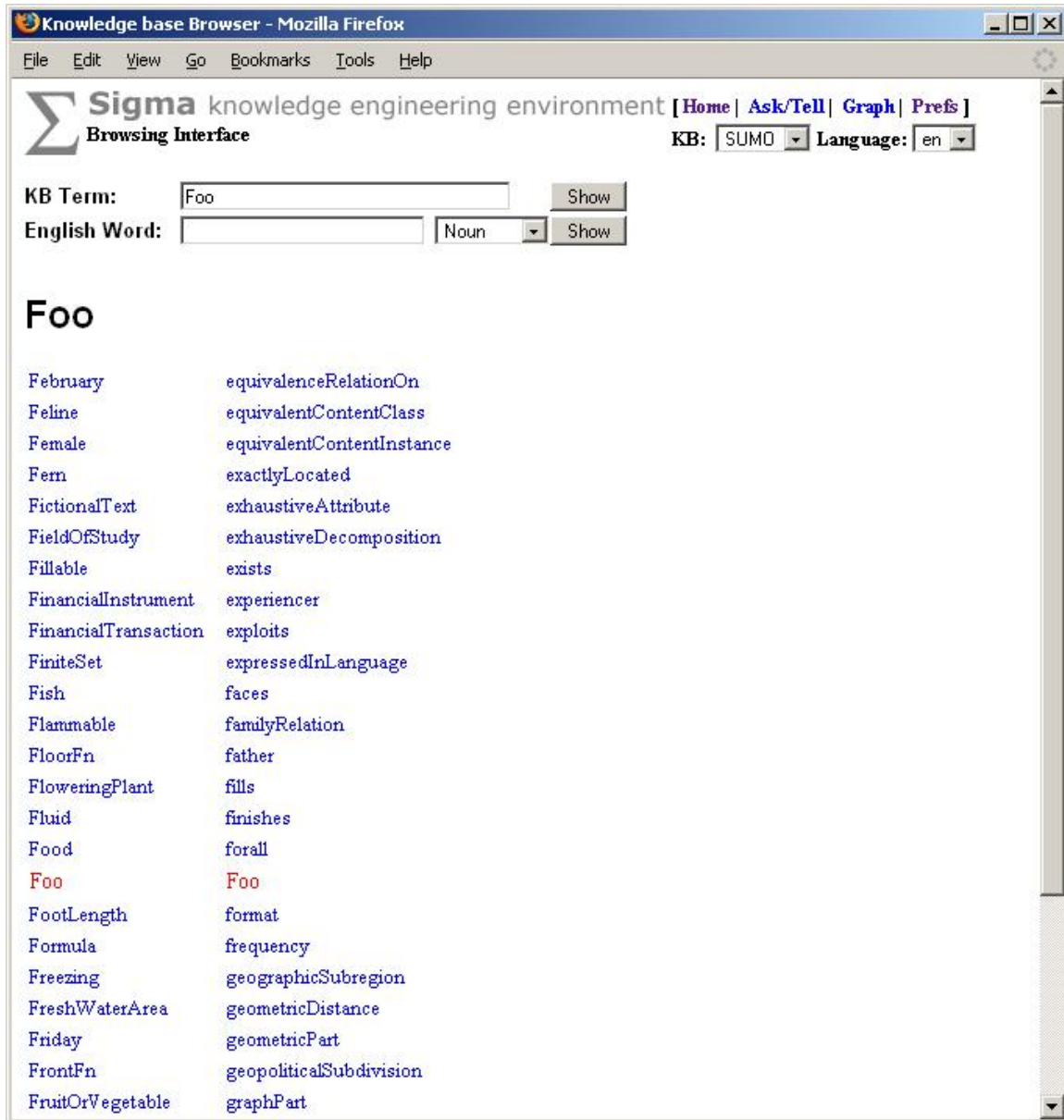


Figure 8: Term neighbors

statement is shown in the right-hand column. The paraphrases are generated automatically from a set of format statements. On the Manifest page (Figure 5), we can see that the file *english_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 pull-down menu labeled “Language”. That menu is constructed automatically, based on the presence of language formatting statements in the knowledge base. Format files for SUMO are currently available in Italian, German, Hindi, and a few other languages.



Figure 9: English word listing

If the user types a word into the KB term box that cannot be found, Sigma responds with a set of terms which are closest, alphabetically, to the given term, as shown in Figure 8.

By entering a word in the type-in area labelled “English Word”, one can get a list of all the matching English word senses in the WordNet lexicon and their mappings to terms in SUMO (Niles & Pease, 2003). The result of entering the word “buffalo”, for example, is the page shown in Figure 9.

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

Another important functionality in Sigma is logical inference over KB contents. Sigma is integrated with the Vampire theorem prover (Riazanov & Voronkov, 2002). To explore this functionality, go to the Knowledge Bases screen and click on the link labeled “Ask/Tell”. For a trivial inference, we can ask for a subclass of the class Entity by posing this query: `(subclass ?X Entity)`. The inference engine returns a very simple proof that the class Abstract is a subclass of Entity, as shown in Figure 10.

While full discussion of the proofs resulting from a resolution theorem prover is 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. The justification can be a list of numbers. The value in the justification column for step 4 shows that it was derived from step 3. Steps can also be taken directly from the knowledge base, which is denoted by “[KB]”, or from the query itself. Currently, no further justification about the inference rule applied is provided, although in this case, one can see that step 4 is derived from step 3 simply by renaming the variable. 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]” as the justification for step 3.

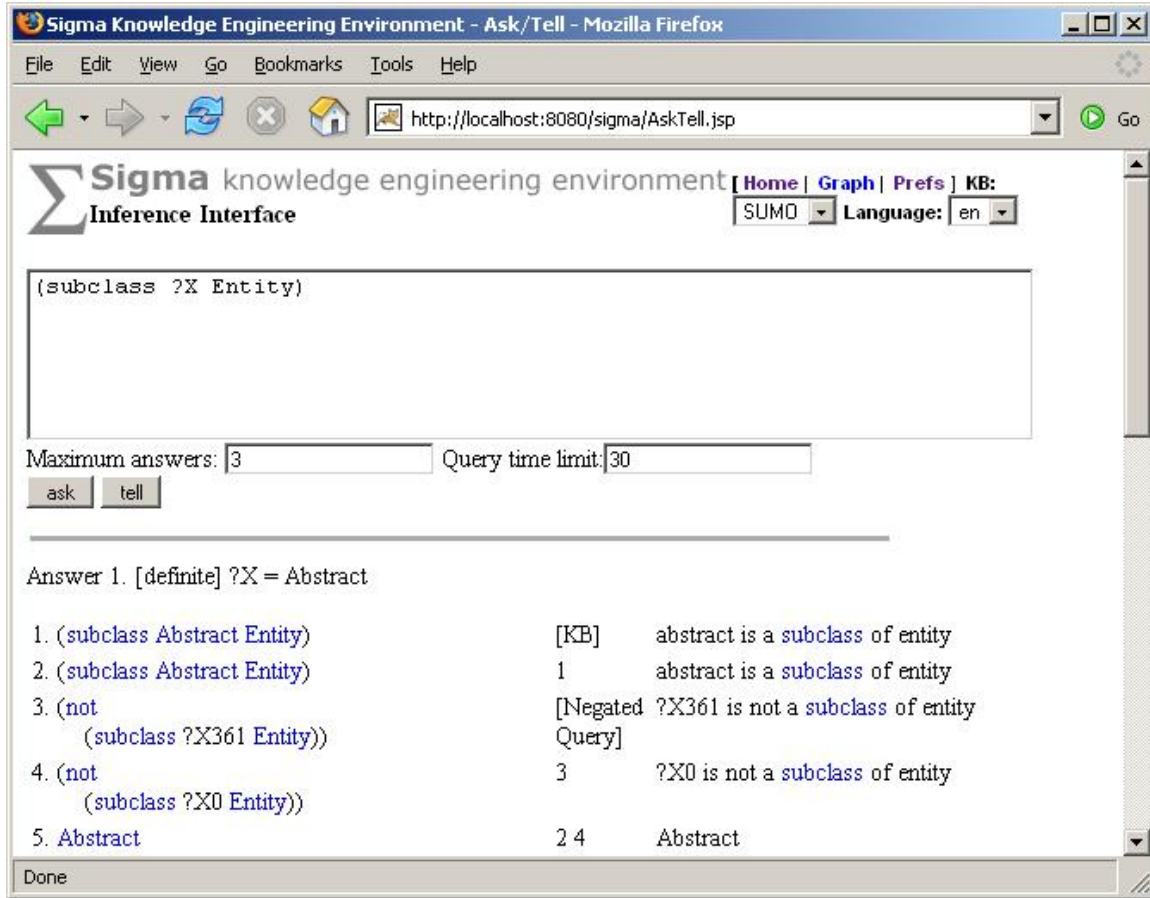


Figure 10: Sigma proof

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 were not provided.

One can also assert an individual formula to the knowledge base by entering a KIF statement, and clicking the button labelled “tell”. The asserted formula then becomes accessible for inference and browsing. If the user has not previously done a “tell” to the current KB, the system creates a new file called <KB-name>_userAssertions.kif, and adds 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 manifest 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 that, although logically true, may not be useful or expected. A final caution on inference: 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.

The screenshot shows a web browser window titled "Sigma Knowledge Engineering Environment - Mozilla Firefox". The address bar shows "http://localhost:8080/sigma/Properties.jsp". The page content includes a header with the Sigma logo and the text "Sigma knowledge engineering environment" and a "[Home]" link. Below the header is a form with several input fields and labels:

- Input field: "C:\projects\articulate\sigma\kif.exe" Label: "Fully qualified path and name of the inference engine"
- Input field: "" Label: "Directory in which the CELT system is located"
- Input field: "" Label: "Fully qualified path and name of SWI prolog"
- Input field: "localhost" Label: "DNS address of the computer on which Sigma is hosted"
- Input field: "8080" Label: "Port number on which Tomcat responds"
- Input field: "SUMO" Label: "Name of the SUMO KB in Sigma"
- Input field: "C:\Program" Label: "Directory in which tests for the inference engine are found"
- Input field: "" Label: "Command to invoke text editor"
- Input field: "" Label: "Command line option for text editor to set cursor at a particular line"
- Radio buttons: "yes" and "no" (selected) Label: "Should caching be employed"
- Radio buttons: "yes" (selected) and "no" Label: "Should cached statements be shown in the term browser"
- Radio buttons: "yes" and "no" (selected) Label: "Should CELT be loaded at startup"
- Button: "Submit Query"

The status bar at the bottom of the browser window shows "Done".

Figure 11: Preferences page

Sigma has a simple caching function that may improve many inferences, since reasoning about subclass relationships is often necessary. Sigma can compute the transitive closure of subclasses statements and assert them directly, so that Vampire does not have to apply the subclass reasoning axiom in SUMO during inference, but can find

them as ground assertions. 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, spend all of its allotted time exploring unhelpful search paths, since it does not know what common-sense answer we are looking for. By asserting directly that `Human` is a subclass of `Object`, we can short-circuit a number of spurious inference paths, and get better and faster results. Caching is turned on from the Preferences page (Figure 11), accessible via the “[Prefs]” link.

The Preferences page also allows the user to set the directory in which the inference engine executable is found, the address of the server 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 (CELT; Pease & Murray, 2003) system should be loaded at startup. The CELT component is experimental, and not generally included in the Sigma distribution.

4 Reference Guide

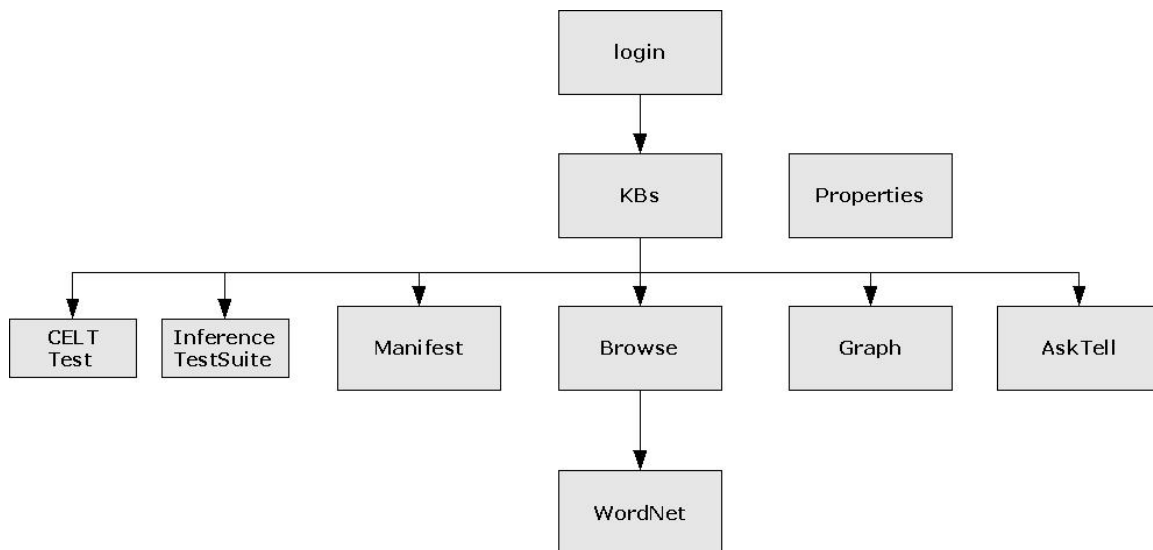


Figure 12: Pages and navigation in Sigma

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

login

This page allows the user to type in a user name 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 the following set of functions:

- *Manifest* – takes the user to the Manifest page, which shows all the files that make up 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; and
- *Remove* – deletes the given knowledge base.

WordNet

This page shows the WordNet synsets for the selected word and part of speech. 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 $(\Rightarrow A \text{ (and } B \text{ } C))$ as one statement and $(\Rightarrow A \text{ } B) (\Rightarrow B \text{ } C)$ as two statements, even though they are logically equivalent. “Total Rules” is also somewhat of an approximation. Sigma counts occurrences of “ \Rightarrow ” and “ \Leftarrow ”. Since $(\Rightarrow$

$A \vee B$ is equivalent to $(\neg A \vee B)$, 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, the term browser page for the term is displayed.

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 Bases page, the Ask/Tell page, the Graph page, and the Preferences 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 assistance to users who are either not comfortable in logic, or are not comfortable in the human language in 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 whether caching should be performed, and whether automatically cached statements should be displayed in the browser. Caching results in storage of the transitive closure of all subclass statements. Depending on the size of the KB, this can be huge amount of information which, although helpful to inference engine performance, is not helpful when shown in the browser.

The user can set whether the CELT (Pease & Murray, 2003) system should be loaded at startup. Starting this component can take several minutes, so it is 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`, 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-name>_userAssertions.kif`, where `<KB-name>` 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 causes a query to be 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.

InferenceTestSuite

This page will run and show the results of inference tests in the directory indicated in the Preferences page. Note that all tests will be completed 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:

- *Terms without documentation* – whether each term occurs in a statement formed with the `documentation` predicate;
- *Terms without a parent* – whether each term occurs in a statement formed with the `instance` or `subclass` predicates; and
- *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.

5 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.

6 Acknowledgements

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

7 Appendix: Natural Language Format

The predicate `format` associates a concept (either a relation or a function) with a string. `format` takes three arguments: the name or abbreviation of a natural language, the relation name and the format string. When there is a need to visualize a 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 is as follows:

- **&%token** – specifies a token that will be made into a hypertext link to the 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 an 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}** – will be 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 representations of a subset of arguments; **range** specifies which arguments will be included; it is a comma separated list of numbers or ranges, for example, “1-4,6” denotes the 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 omitted, in which case **{range}** defaults to all arguments, and **[delim]** defaults to a single space.
- **%%** – will be replaced with a single percent character.

The predicate `termFormat` 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.

8 Appendix: Test Formats

Inference test files are legal KIF files. They must end with the extension “.tq”. Inference tests support several special purpose predicates. “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 composed of a variable name and a 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 as an inference test. It must be named `celtTest.txt`. It must be legal KIF. It should consist of 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.