Sigma Programmer's Manual

Adam Pease
3/8/04

Introduction

Sigma (Pease, 2003) is an environment for creating, testing, modifying and inferencing on ontologies. The most detailed and current programmer's documentation on Sigma is found in its JavaDoc formatted comments. This document provides a brief, and more graphical overview of the system, but the programmer should rely primarily on the JavaDoc. Programmers should also refer to the Sigma Users' Guide and SUO-KIF manual.

Directory Structure

Sigma has a fairly simple directory structure. The Vampire and CELT directories are set in the config.txt file, and can be changed by editing the file directly, or from the Sigma Preferences.jsp page. Note that CELT is not generally distributed with Sigma because it is in an experimental stage of development.

[Tomcat]

- webapps/sigma – location of the main Sigma code
- KBs – location of all knowledge bases, preprocessed knowledge bases, NL format files and config.txt configuration file
- tests – location of all test files

Vampire – location of the kif.exe inference engine file

CELT – location of the language to logic translation code

SWI-Prolog – location of the plwin.exe prolog executable

Code Structure

The primary class for Sigma is **`KBmanager`**. It contains an instance of all the in-memory content of Sigma. It managers the config.txt configuration file and all the knowledge bases. Beneath **`KBmanager`** are instances of **`KB`**. Each KB contains all the content of the files in that particular knowledge base, as well as instances of the Vampire and CELT processes. A KB depends on the class **`KIF`** to read and write knowledge base files. **`KIF`** in turn depends on **`Formula`** to handle operations on individual formulas. **`Formula`** also takes care of pretty-printing formulas with proper indenting and line feeds. **`Formula`** also processes statements for use in the class **`Vampire`**. **`Vampire`** communicates with the rest of Sigma through an XML interface. **`BasicXMLparser`** supports communicating through this interface, along with its subsidiary class of **`BasicXMLelement`**. Results returned from Vampire, after being parsed with **`BasicXMLparser`**, are further processed with **`ProofProcessor`** and **`ProofStep`**. Proof results, as well as KB contents, are displayed on jsp pages with the help of **`HTMLfomatter`**, which takes care of creating the HTML links for terms in a formula.

Compilation

Sigma uses Ant (like Unix make) for its build system. build.xml contains information on directory structures for building and deploying the Sigma system. build.xml will set up some directories under itself for the output of the compilation process. The only item inside build.xml that the developer should have to change is the directory in the element dist.home (line 8 in the file). Currently, the only supported version of the Java environment needed for Sigma is J2SE Java 2 SDK v1.4.1_05 and the Tomcat v4.0.4 JSP server.

The recommended compiler for the Vampire C++ code on Linux is gcc 2.95.3. Vampire compiles under Microsoft Visual C++ version 6 on Windows. VC++ project files are included in the Sigma distribution in /vampire. It should be possible to use the supplied kif.exe file without recompilation however. A makefile for compilation on Unix/Linux is also provided as /vampire/makefile

Natural Language Paraphrase System

The Sigma NL paraphrase system depends on the existence of the "language.txt" file in the KBs directory, plus existence of at least one language template file, such as "english-format.kif". The class **`NLformatter`** handles this work.

Inference Engine Communication

Vampire.java is the class for invoking the KIF version of Vampire from Java. Vampire takes XML input and returns XML output. Input forms are either **`<query optionalArgs>`**KIF formula**`</query>`** or **`<assertion>`**KIF formula**`</assertion>`**. Results of queries have the following form:

```
<queryResponse>
  <answer=yes/no number ='#'>
    <bindingSet type='definite/disjunctive'>
      <binding>
        <var name='' value=''>
      </binding>
    </bindingSet>
    <proof>
      <proofStep>
        <premises>
          <premise>
            <clause/formula number='#'>
              KIF formula
            </clause>
          </premise>
        </premises>
        <conclusion>
          <clause/formula number='#'>
            KIF formula
          </clause>
        </conclusion>
      </proofStep>
    </proof>
  </answer>
  <summary proofs='#'>
</queryResponse>
```

Note that if the result of a query is not a variable binding, then the **<bindingSet>** element will be omitted.

Vampire requires a number of modifications to KIF knowledge bases to ensure that they are in a more explicit first-order format. This processing includes prepending a 'holds' predicate before every non-logical predicate, quoting formulas that are arguments to non-logical predicates, expanding row variables by treating them as a macros, and translating arithmetic operators. This translated KIF is placed in the KBs directory with the name of the current knowledge base and "-v.kif" appended to it. For example, a knowledge base called "SUMO" would be saved as "SUMO-v.kif"

Orphaned Code

KIFplus is code that is intended to be a full KIF parser with proper error checking. It is not complete and is not currently used.

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.