From: AAAI Technical Report WS-96-05. Compilation copyright © 1996, AAAI (www.aaai.org). All rights reserved.

An HTML Interface for Classic

Christopher A. Welty
Vassar College Computer Science Dept.
Poughkeepsie, NY 12601
weltyc@cs.vassar.edu
`http://www.cs.vassar.edu/faculty/welty`

Abstract

Several significant problems exist when applying knowledge representation systems to real problems. In particular, the obscurity of LISP, the resource consumption, garbage collection, and the single user nature of most KR systems can prevent the technology from being accepted in many non-research environments. This paper briefly presents one way to address these problems with a web interface, and outlines some of the general principles that guided the implementation of a web interface for Classic.

1 Introduction

Implemented Knowledge Representation systems have much to offer in solving serious problems in the real world. In pursuing this belief, we have gathered experience transferring KR technology to users outside traditional academic circles. Most of this work has been in the context of AI & Software Engineering [Welty, 1995a; 1995b], but a recent project in building a card catalog system for the web [Welty, 1996] has led to some new web-based solutions for the problems facing this type of technology transfer.

This research has made use of Classic [Brachman *et al.*, 1991] as the vehicle for knowledge representation. The paper begins with a description of the barriers that were met when trying to use Classic to solve some knowledge problems in a small software company. The paper then discusses the web interface that was devolped to address these problems and a classic-based card catalog system for the web.

2 Barriers

The principle barriers to deploying Classic in industry are mainly related to its status as a tool for KR researchers. Specifically, in an attempt to track knowledge in a small software company related to their main product, the use of Classic was rejected for many reasons:

- Resources: The memory and processing requirements of a machine using Classic are high. The company could not afford such machines.
- Multiple Users: Classic is a single user system. The knowledge to be represented was needed across the company, and frequently simulatneous access was required.
- LISP: None of the programmers in the company knew LISP, and management did not like the parenthesis.
- GC: Several managers in the company had heard that LISP has a problem called garbage collection. When the system went into GC, they noticed.

In the end, we used an Oracle application to represent the knowledge this company needed.

3 CL-HTTP

In general, just about any user interface to Classic would solve some of the problems mentioned in the previous section, but an HTML interface can address each of the barriers listed above:

- Resources: Only one machine is required that has the resources to run Classic. Anything can be a web client.
- Multiple Users: The web interface allows multiple simultaneous http "hits".
- LISP: The syntax of LISP and Classic are hidden by the interface.
- GC: When the server starts garbage collecting, the web browser gives the same response it would when experiencing a net hang. Managers don't seem to have a problem with net hangs.

The technology that made this possible was the Common LISP Hypermedia Server (CL-HTTP) [Mallery, 1994]. This HTTP server runs in most multi-threaded Common LISPs, and allows for active URLs to be served by LISP functions (as opposed to shell or Perl scripts). The real advantage of this is that since the server can run as a thread within the same application as Classic, it shares memory with it and no inter-process communication is necessary for the HTTP response functions to access the knowledge-base. This *greatly* simplified the task of creating this interface.

4 The Interface

While the technology the web represents was clearly dated from its inception, its familiarity simply can not be taken for granted - everyone knows how to use a web browser, and navigate through hypertext documents.

That there is an analogy between semantic networks and hypertext should be clear upon the face of it, and it was this obvious connection that first made an HTML interface seem like a natural way to describe Classic objects. The interface has four basic modes: describing concepts, describing individuals, listing taxonomies, and entering queries.

It is important to realize that all of these pages are generated *on the fly*, that is they always reflect the current state of the knowledge base.

4.1 Describing Concepts

When describing a concept, the interface displays a web page whose name is the concept name, underneath which is the comment for that concept. If the concept has any individuals, a link is generated that when clicked on will produce a page full of links to descriptions of those individuals.

The concept's direct parents are listed by name, with each name also being a link to the description of that concept. The list of parents is followed by a list of all the ancestors, in no particular order. This is followed by a list of child concepts, followed by descendants.

Finally, the role restrictions for the concept are listed, by role. For simplicity, the interface does not display roles for which there is an `at-most 0` restriction, as these tend to clutter the view. Eventually this will be generalized to obey the *interesting* meta-data facilty provided in the latest version of Classic.

The roles are listed in an HTML unordered (`<UL></UL>`) list, and the role hierarchy is displayed as nested lists below the parent role.

Any reference to another concept (such as within an `ALL` restriction) is, of course, a link to a description for that concept.

All derived information is display in italics.

A simple example of the text in a concept description page is shown below.

BOOK

A book is a collection of chapters. It has an author and a title.

Parents: Publication.
Ancestors: Publication, *Classic-Thing, Thing.*
Children: *Manual*
Descendants: *Manual*

Click here for a list of individuals

Title: All STRING, [1,1]
Author: All PERSON, [1,]
Publisher: All PUBLISHER
Data-role: no restrictions
Date: All INTEGER, [1,1]

4.2 Describing Individuals

Individuals in Classic are very similar to concepts, and as a result the description pages of individuals look very similar. There are a few differences, some of which are based on the way Classic treats individuals, and some were based on the way our system used them.

From a Classic perspective, individuals can not have individuals, so that section in the concept descriptions was removed. Individuals can also have *closed* roles, and those were displayed in bold face.

For the purposes of our card catalog system, which was the motivating force for creating the interface, individuals were the key element. That is, users frequently did not care about concepts, and browsed almost exclusively individuals. The information provided in these descriptions, then, had to be tailored based on people's existing experiences with card catalogs and the new functionality offered by Classic.

To begin with, displaying all the information Classic maintains about an individuals was out of the question. The page quickly becomes cluttered and unusable. The interface was created before the *interesting* meta-data facilty was added to Classic, and so it made use of certain assumptions based, again, on our use of the system.

- In the card catalog system, the only relevant role restriction is the `fills` restriction. Therefore roles that had no fillers were not displayed.
- The role hierarchy was mostly supressed. For every filler, only the most specific role with that filler was displayed. If a parent role contained a filler that was not contained by any of its descendent roles, then the role hierarchy was displayed below that role (as nested HTML lists), only if the descendant roles had fillers.
- Every individual had a *key role.* This role was identified in the meta-individual of the individual's direct parent. The value of this role was assumed to be a string, and was used in conjunction with the Classic name of the individual when a one-line reference to that individual was needed (such as when used as a filler for a role in another individual). Without the key-role facility, a page describing an individual of the concept `book`, for example, might have a role called `author` filled with the individual `person-10`. Even though the reference to `person-10` would be a link that would provide a description of that individual, it is too much work and too unnatural for a card catalog user to be expected to do it. Here we had a case where Classic naturally provided too little information, and the interface needed to augment it.

The key role was also always the first role displayed in the full description.

- Every individual had *important roles.* These were listed also as meta-information, and identified roles that were displayed after the key role in the full description, but before any others. For individuals of `book`, for example, have `title` as a key role, and `author`, `publisher`, and `date` as important roles. This puts the most relevant information for the user at the top.
- Every individual had *hidden roles.* These were also listed as meta-information, and identified roles that by default should not be displayed.

  Important roles and hidden roles are accounted for in the *interesting* facility added to Classic in version 2.3.

Many of the compromises made above for displaying individuals seem to dilute the description logic nature of Classic, in particular the fact that only role fillers are displayed for individuals. It is important to realize that this is only the default behavior, future versions of the interface will be easily customizable for users who are familiar with Classic.

An example of an individual description is shown below.

BOOK-12: The Great Book

Parents: Book.
Ancestors: Publication, *Classic-Thing, Thing.*

Title: The Great Book
Author: [PERSON-10: Chris Welty]
Publisher: [ORGANIZATION-14: ACM]
Date: 1996

4.3 Listing Taxonomies

Again, the attraction of the CL-HTTP server was that very little programming was required to produce a fairly usable interface to Classic. Graphical views, though clearly desirable particularly when viewing taxonomies, were ruled out due to their inherent complexity.

We chose to view taxonomies as HTML unordered lists. Given a concept, the children below that concept are displayed as items with their comments. Descendants are displayed in nested lists. In addition, two links are provided per concept, one (expand) allows the user to make that concept the root concept of the taxonomy being viewed. The other provides the full concept description.

Finally, each taxonomy page has a link at the top that can restrict the depth of the taxonomy being displayed. This feature helps viewing deep taxonomies, and when combined with the expand link makes it very easy to browse taxonomies.

4.4 Queries

The primary advantage of using Classic in the card catalog domain was the ability to use more expressive queries. The interface is still under development, however, and this aspect is still primitive. To enter a query, you type a classic concept expression in an HTML form, and the interface will display a list of all the individuals that are subsumed by that concept.

Clearly this aspect of the interface needs to be extended.

5 Results and Conclusion

A demo of the web interface is accessible through my home page, under the link for the Untangle project. The Untangle project is the library card catalog system currently under development.

The interface has worked well in helping undergraduates understand the ontology and how to enter data. These undergraduates, it should be noted, have little or no KR background, and certainly no experience with description logics, and are able to navigate through the knowledge-base with fairly little training. In this respect, the interface works well.

The barriers to introducing Classic to a small software company were addressed, and initial feedback from that company indicates they are interested in seeing it.

The future use of description logics such as Classic will depend on the barriers discussed being addressed in such a way.

References

[Brachman *et al.*, 1991] Ronald J. Brachman, Deborah McGuinness, Peter Patel-Schneider, Alex Borgida, and Lori Resnick. Living with classic: When and how to use a kl-one-like language. In *Principles of Semantic Networks*, pages 401–456. Morgan Kaufman, May 1991.

[Mallery, 1994] John A. Mallery. A common lisp hypermedia server. In *Proceedings of The First International Conference on The World-Wide Web*, Geneva, Switz., May 1994. CERN.

[Welty, 1995a] Christopher A. Welty. *An Integrated Representation for Software Development and Discovery.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180, 1995.

[Welty, 1995b] Christopher A. Welty. Towards an epistemology for software representations. In Dottie Setliff, editor, *Proceedings of KBSE-95, The Tenth Knowledge-Based Software Engineering Conference*, pages 148–154. IEEE CS Press, November 1995.

[Welty, 1996] Christopher A. Welty. Intelligent assistance for navigating the web. In John Stewman, editor, *Proceedings of FLAIRS-96, The Florida AI Research Symposium*, pages 311–315, May 1996.