Knowledge Retrieval and the World Wide Web

Philippe Martin and Peter W. Eklund, Griffith University

THE WEB IS CURRENTLY A DISTRIBUTED MASS OF SIMPLE HYPERTEXT DOCUMENTS. LARGE-SCALE WEB SEARCH ENGINES SUCH AS ALTAVISTA AND INFOSEEK ARE NOT CAPABLE OF RETRIEVING PRECISE INFORMATION RESULTS. THE AUTHORS PRESENT A NEW TOOL, WEBKB, THAT INTERPRETS SEMANTIC STATEMENTS STORED IN WEB-ACCESSIBLE DOCUMENTS.

LARGE-SCALE WEB SEARCH engines effectively retrieve entire documents, but they are imprecise, because they do not exploit and hence retrieve the semantic Web document content. We cannot automatically extract such content from general documents yet. Manually structuring Web documents—for example, with XML—lets us retrieve more precise information using string- and structure-matching tools, such as the Web robots Harvest, WebSQL, and WebLog. However, this approach is not scalable, because it only retrieves fine-grained information if the documents are thinly structured and the querier knows their structures, exact names, and forms.

Knowledge representation languages that support logic inference can help us achieve more flexible and precise knowledge representation and retrieval. Industry is currently developing many metadata languages to let people index Web information resources with knowledge representations (logical statements) and store them in Web documents. However, these metadata languages are insufficient to satisfy several requirements necessary to allow precise, flexible, and scalable information retrieval.

On the basis of ease and representational completeness, we argue in favor of general and intuitive knowledge representation languages such as conceptual graphs (CGs)[1] rather than the direct use of XML-based languages. To let users represent knowledge at the level of detail they require, we propose simple notations for restricted knowledge representation cases and a technique that lets users leave knowledge terms undeclared. We built a Web-accessible tool (CGI server), WebKB,[2,3] to support this approach and let its users combine lexical, structural, and knowledge-based techniques to exploit or generate Web documents. WebKB is an ontology server and directed Web robot. (See the sidebar for a list of related URLs.)

Metadata language requirements

A first requirement is that the metadata language must be intuitive and concise enough for people to use easily (after a short training period). Most current knowledge-oriented metadata languages are built above XML, such as the Resource Description Framework (RDF) and Ontology Markup Language (OML). The choice of XML as an underlying format lets you use standard XML tools to exchange and parse these metadata languages. However, because XML is verbose, the metadata languages built above it are verbose and difficult to use without specialized editors. Such editors do not eliminate the need for people to use a language to represent knowledge (except in application-dependent editors that only let you fill predefined "frames"). Consequently, with XML-based languages, we must write information in two versions—one for machines and another for humans.[4] Additionally, standard XML tools are of little use, because we still require specialized editors,

1094-7167/00/$10.00 © 2000 IEEE

Relevant URLs

AI-Trader	www.vsb.informatik.uni-frankfurt.de/projects/aitrader/intro.html
Apecks	ksi.cpsc.ucalgary.ca/KAW/KAW98/tennison/index.html
CGI server	www.w3.org/CGI
Co_4	ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html
Conceptual Graphs	meganesia.int.gu.edu.au/~phmartin/WebKB/doc/CGs.html
Harvest	ww.ncsa.uiuc.edu/SDG/IT94/proceedings/searching/schwartz.harvest/schwartz.harvest.html
Knowledge Interchange Format	logic.stanford.edu/kif/kif.html
OML	www.ncgr.org/research/genex/ontology.html
Ontobroker	ontobroker.aifb.uni-karlsruhe.de
Ontolingua ontology server	www-KSL-SVC.stanford.edu:5915
Ontosaurus	www.isi.edu/isd/ontosaurus.html
Resource Description Framework	www.w3.org/RDF
Shore	www.cs.wisc.edu/shore
Tadzebao	ksi.cpsc.ucalgary.ca:80/KAW/KAW98/domingue
Visual languages	www.cs.orst.edu/~burnett/vpl.html
WebKB	meganesia.int.gu.edu.au/~phmartin/WebKB
WebLog	www.cs.concordia.ca/~special/bibdb/weblog.html
WordNet	www.cogsci.princeton.edu/~wn
Word Wide Web Consortium	www.w3.org
XML	www.w3.org/XML

analyzers, and inference engines to manage these languages.

To reduce information redundancy, Ontobroker, an ontology that guides information retrieval from annotated HTML documents accessible on the Web, provides a notation for embedding attribute-value pairs inside an HTML hyperlink tag. A document's author can use these tags to delimit an element. Thus, he or she can implicitly reference each element in the knowledge statement within the tag enclosing the element. A wrapper has been added to Ontobroker to generate RDF definitions automatically from Ontobroker metadata.

Along this same line, a document's author should be allowed to make some knowledge statements visible to readers. This is an obvious requirement when we can use an especially intuitive notation—for example, when we can make graphics with a visual language or write sentences with a *controlled language*, a subset of natural language that eliminates ambiguity. This visualization feature is also handy with any notation when the document provides explanations about the knowledge statements it stores. In this way, for example, we can integrate a knowledge base and its associated documentation within the same document and access both using classic searches (such as string-matching, navigating a table of contents, and so on) and knowledge-based searches.

Although the Ontobroker metadata language was designed to reduce information redundancy, users cannot show statements, because they are within HTML tags. Furthermore, like the RDF, the Ontobroker metadata language is essentially a notation for attribute-value pairs. Such a representation is general but basic and hard to read. Finally, only the document authors can index any of its parts because they index document elements with HTML tags. Others are limited to only those elements that are accessible via URLs.

A metadata language should also be sufficiently precise and general enough to let users represent any Web-accessible information at the desired level of precision. This implies that the metadata language is based on an expressive formal model and that it has a notation letting the user exploit the formal model's expressivity. Any formalism equivalent to first-order logic that permits the use of contexts is an appropriate candidate. Knowledge Interchange Format (KIF) and CGs are good examples. It is important not to restrict users, but for efficiency reasons, a search engine can ignore some features in knowledge statements. For example, a CG-based search engine can ignore references to sets within CGs and still exhibit adequate precision (the CGs with references to sets are also retrieved). The Ontobroker metadata language and the RDF are general but imprecise, because they are oriented toward representing entire documents (not arbitrary parts of them) and do not propose conventions to represent logic-based features, such as quantifiers and operators. This limits the capacity of their statements.

WebKB

The three first requirements for precise, flexible, and scalable information retrieval imply several easy to use notations (some intuitive, some precise and expressive) and the possibilities to insert them anywhere in a Web document. WebKB interprets chunks of knowledge statements in Web documents to satisfy these requirements. Two special HTML marks (`<KR>` and `</KR>`) or the strings `$(` and `)$` must delimit each chunk, or group of statements. These chunks are visible unless the document's author hides them with HTML comment tags. The author must specify the knowledge representation language used in each chunk at its beginning: `<KR language="CG">`.

Currently, WebKB can interpret the linear notation of CGs and morereadable notations we have invented: a formalized English and a frame-like CG linear notation. These simpler notations are automatically translated into CGs. We chose the CG formalism, first, because it has a graphical and linear notation that are both concise and easily comprehensible. Second, we can reuse two CG inference engines—CoGITo[5] and Peirce[6]—that exploit subsumption relations defined between formal terms for calculating specialization relations between graphs and therefore between a query and facts in a knowledge base. Hence, we can make statements and queries at different levels of granularity.

Another requirement is that each user should not have to explicitly declare and organize all the terms in the knowledge statements. Indeed, declaring and organizing terms is a tedious and often complex work that detours most users and is probably one of the main reasons why so few hypertext systems have been knowledge-based (MacWeb[7] is an exception). This requirement is a rationale for semiformal knowledge-representation languages such as concept maps (sern.ucalgary.ca/~kremer/tutorials/conceptmaps/high), as opposed to logic-based formalisms such as KIF. The use of semi-formal statements is at the expense of knowledge precision and accessibility but allows rapid expression and incremental refinement of knowledge. When forewarned by a special command (`no decl`), WebKB accepts CGs that include some undeclared terms. Another informal feature WebKB accepts are notations for sets within CGs: WebKB ignores them during searches but displays each retrieved CG in the

Figure 1. The WebKB tool menu and a knowledge-based information retrieval and handling tool. The example query shows how a document containing indexing images is loaded by the browser into the WebKB processor and then how the command `spec`, which looks for specializations of a conceptual graph, can be used to retrieve CGs and the images they index. According to the value selected in the "kinds of results" option (top right), the images, but not the knowledge statements, will be presented.

form in which it was entered.

HTML and XML do not let users reference or index any part of a document that they have not created. WebKB provides an indexation notation that lets a document element be referred by its content or occurrence in a document.

Simply representing knowledge within documents is insufficient; knowledge- and string-based commands are also necessary. It is handy to be able to use them within documents and—if desired—have the results automatically inserted in place of the commands. The hypertext literature refers to this technique as *dynamic linking* and calls the generated document a dynamic or *virtual document*.[7] This idea has many applications—for example, adapting a document's content to a user. A procedural or declarative language should combine the commands and their results. Web robots perform some document generation in that way but current metadata languages only allow knowledge representation. WebKB lets users generate virtual documents and combine lexical, structural, and knowledge-based data management by proposing commands for searching and joining CGs, Unix-like file management commands working on Web-accessible documents, and a simple Unix shell-like script language to combine commands. Users can insert these commands in documents or in form-based interfaces. Figure 1 shows the WebKB tool menu and the knowledge-based information retrieval and handling tool, the main general interface to WebKB.

Although this document-based approach is handy, its scalability is limited. For example, before using knowledge query commands, the WebKB user must either directly assert some knowledge or use loading commands (such as `load URL`) to specify Web documents that include the knowledge being exploited. To let users benefit from the knowledge of users they do not know, we are currently extending WebKB to handle a cooperatively built knowledge repository.

Knowledge representation languages versus XML-based metadata languages

To represent knowledge within documents, we advocate using knowledge representation languages over XML-based metadata languages. To compare the alternatives, Figure 2 shows how to represent a simple sentence with CGs in WebKB and with KIF and the RDF. The sentence is, "John believes that Mary has a cousin who is her age."

The CG representation in Figure 2 seems simpler than the others. The semantic network structure of CGs has three advantages:

- it restricts knowledge formation without compromising expressivity, which tends to computationally ease knowledge comparison;
- it encourages users to fully describe relations between concepts (for instance, as opposed to languages where they can use "slots" of frames or objects); and
- it permits a better visualization of relations between concepts.

Even if CGs seem relatively intuitive, they are not readable by everyone. In restricted cases, we might prefer simpler notations. For instance, Figure 3 shows notations that WebKB accepts as equivalent to the following CG:

```
TC for KADS_conceptual_model(x)
  are  //note: TC means "Typical
         Conditions"
[KADS_conceptual_model:*x]-
{  (Part)->[Model_of_problem_
     solving_expertise];
   (Part)->[Model_of_
     communication_expertise];
   (Part)->[Model_of_
     cooperation_expertise];
   (Input)<-[Knowledge_ design]-
     >(Output)->[Knowledge_
     base_system];
}
```

Undeclared terms in knowledge statements

Users might not want to take the time to declare and order most of the terms they use when representing knowledge. For example, this might be the case when a user indexes sentences from various documents for private knowledge organization purposes.

```
<KR language="CG">
load "http://www.bar.com/topLevelOntology"; //Import this ontology
Age < Property; //Declare Age as a subtype of Property
Cousin(Person,Person) {Relation type Cousin};
[Person:"John"]<-(Believer)<-[Descr: [Person:"Mary"]- { (Chrc)->[Age: *a];
                                                       (Cousin)->[Person]->(Chrc)->[*a];
                                                     } ];
</KR>
```

```
<KR language="KIF">
load "http://www.bar.com/topLevelOntology";   //Import this ontology
(Define-Ontology Example (Slot-Constraint-Sugar topLevelOntology))
(Define-Class Age (?X) :Def (Property ?X))
(Define-Relation Cousin(?s ?p) :Def (And (Person ?s) (Person ?p)))
(Exists ((?j Person))
   (And (Name ?j John) (Believer ?j '(Exists ((?m Person) (?p Person) (?a Age))
                                       (And (Name ?m Mary) (Chrc ?m ?a)
                                            (Cousin ?m ?p) (Chrc ?p ?a)
                                       ))
                        ))) </KR>
```

```
          <!— RDF notation; assumed location: http://www.bar.com/example —>
<RDF xmlns="http://www.w3.org/TR/WD-rdf-schema#"
     xmlns:t="http://www.bar.com/topLevelOntology#">
  <Class ID="Age"><subClassOf resource="t:Property"/></Class>
  <PropertyType ID="Cousin" comment="Relation type Cousin">
                           <range resource="t:Person"/>
                           <domain resource="t:Person"/></PropertyType> </RDF>
<RDF xmlns="http://www.w3.org/TR/WD-rdf-schema#" xmlns:x="http://www.bar.com/example#"
     xmlns:t="http://www.bar.com/topLevelOntology#">
  <t:Person bagID="Statement_01"><t:Name>Mary</t:Name>
                                <t:Chrc><x:Age ID="age"></x:Age></t:Chrc>
                                <x:Cousin><t:Person><t:Chrc resource="x:age"/></t:Cousin>
</t:Person>
<Description aboutEach="#Statement_01" t:Believer="John"/> </RDF>
```

Figure 2. Comparing knowledge representation with the Knowledge Interchange Format, conceptual graphs, and the Resource Description Framework.

```
                        "=>" of a "necessary" relation, "<=" of a sufficient relation) */
KADS1_conceptual_model.
    Part: Model_of_problem_solving_expertise,
          Model_of_communication_expertise,
          Model_of_cooperation_expertise.
    Input of: Knowledge_design (Output: Knowledge_base_system).
```

```
    /* Text structured with HTML tags (and same conventions for relations) */
<dl><dt>KADS1_conceptual_model
    <dd>Part: <ul><li>Model_of_problem_solving_expertise
                  <li>Model_of_communication_expertise
                  <li>Model_of_cooperation_expertise
              </ul>
    <dd>Input of: Knowledge_design (Output: Knowledge_base_system)
</dl>
```

```
      /* Formalized english */
Typically, a KADS1_conceptual_model has for part a model_of_problem_solving_expertise,
a model_of_communication_expertise and a model_of_cooperation_expertise.
Typically, a knowledge design has for input a KADS1_conceptual_model and has for output a
knowledge_base_system.
```

Figure 3. Complementary notations for simple knowledge statements.

```
$(Indexation
   (Context: Language: CG;
             Ontology: http://www.bar.com/topLevelOntology.html;
             Repr_author: phmartin; Creation_date: Mon Sep 14 02:32:21 1998;
              Indexed_doc: http://www.bar.com/example.html; )
   (DE: {2nd occurence} the red damaged vehicle )
   (Repr: [Color: red]<-(Color)<-[Vehicle]->(Attr)->[Damaged] )
)$
$(DEconnection
   (Context: Language: CG;
             Ontology: http://www.bar.com/topLevelOntology.html;
             Repr_author: phmartin; Creation_date: Mon Sep 14 02:53:36 1998;)
   (DE: {Document: http://www.bar.com/example.html} )
   (Relation: Summary)
   (DE: {Document: http//www.bar.com/example.html} {section title: Abstract})
)$
```

Figure 4. A language for knowledge indexing or connecting any Web-accessible document element.

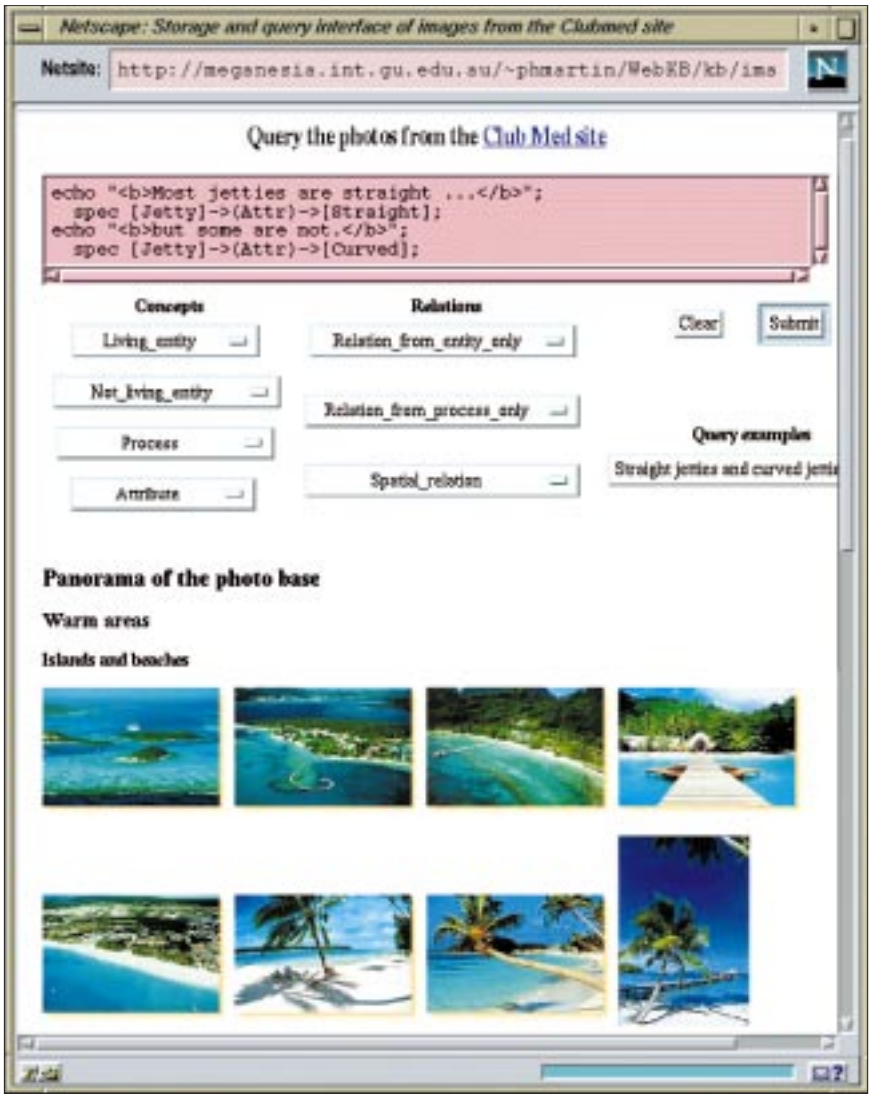

Figure 5. Images, knowledge indexations, and a customized query interface contained within one document. The sample query shows how the command `spec`, which looks for specializations of a conceptual graph, can be used to retrieve images CGs indexed. (Figure 7 gives the results.)

To permit this and still let the system perform some minimal semantic checks and knowledge organization, we propose the casual user represent knowledge with basic declared relation types and leave undeclared the terms used as concept types. This method has four rationales:

Second, if knowledge statements are made from concepts linked by basic relations—that is, if the complexity is manifest within concept types rather than in relation types—only a limited set of relation types are necessary for an application. WebKB already proposes a top-level ontology of 200 basic relation types[8,9] collecting common thematic, mathematical, spatial, temporal, rhetorical, and argumentative relations types.

Second, WebKB can use relation signatures to give suitable types to the undeclared terms used as concept types. For instance, in the top-level ontology WebKB proposes, the relation types `Input`, `Output`, `Agent`, `Method`, `SubProcess`, and `Purpose` are all defined to have a concept of type `Process` as the first argument. Hence, in the previous example, WebKB can infer that `Knowledge_design` must be a subtype of `Process`.

Third, we merged the natural language ontology WordNet—120,000 words linked to 90,000 concept types—into our top-level ontology.[8,9] When users implement and initialize the WebKB shared repository with these ontologies, WebKB will be able to semiautomatically relate the undeclared terms used as concept types to precise concept types in the repository, thanks to links between words and concept types and to constraints imposed by the relation signatures. For example, consider the following CG where users have not declared the terms cat and table:

`[Cat]->(On)->[Table]`. In WordNet, cat has five meanings (feline, gossiper, x-ray, beat, and vomit), and table has five meanings (array, furniture, tableland, food, and postpone). In the WebKB ontology, the relation type `On` connects a concept of type `Spatial_entity` to another concept of the same type. Thus, WebKB can infer that beat and vomit are not the intended meanings for cat, and array and postpone are not the intended meanings for table. To further identify the intended meanings, WebKB could prompt the following questions to the user: "Does cat refer to feline, gossiper, x-ray, or something else?" and "Does table refer to furniture, tableland, food, or something else?"

Finally, knowledge statements are more readily comparable if they follow the same conventions. Thus, the convention of using basic relations is important. (The opposite convention using primitive concepts and complex relations would be much harder to follow). For example, consider the sentence, Mary is 20 years old. Following our conventions it is better to use the concept type `Age ([Person:"Mary"]->(Chrc)->[Age:@20])`, unless a user has predefined this relation type:

```
relation Age (x,y) is [Age]-
  { (Chrc)->[Livingentity:*x];
    (Measure)->[Integer:*y];
  }
```

(This solution implies that the inference engine expands the relation type `Definition` when comparing graphs. Few CG engines can perform type expansion.)

By default, WebKB enforces declared terms in the CG linear notation but permits undeclared terms in simpler notations (see Figure 3). The commands `decl` and `no decl` overide this default mode, and an exclamation mark before a type explicitly tells the system that the type was deliberately left undeclared. We can also use quoted sentences; WebKB understands them as individual concepts of the type `Description`.

Another facility of the WebKB parser is that, like HTML browsers, it ignores HTML tags (except definition list tags) in knowledge statements. However, when these statements are displayed in response to a query, they are displayed using the exact form given by the user, including HTML tags. Thus, the user can combine HTML or XML features with knowledge statements. For example, the user can put some types in italics or make them the source of hypertext links.

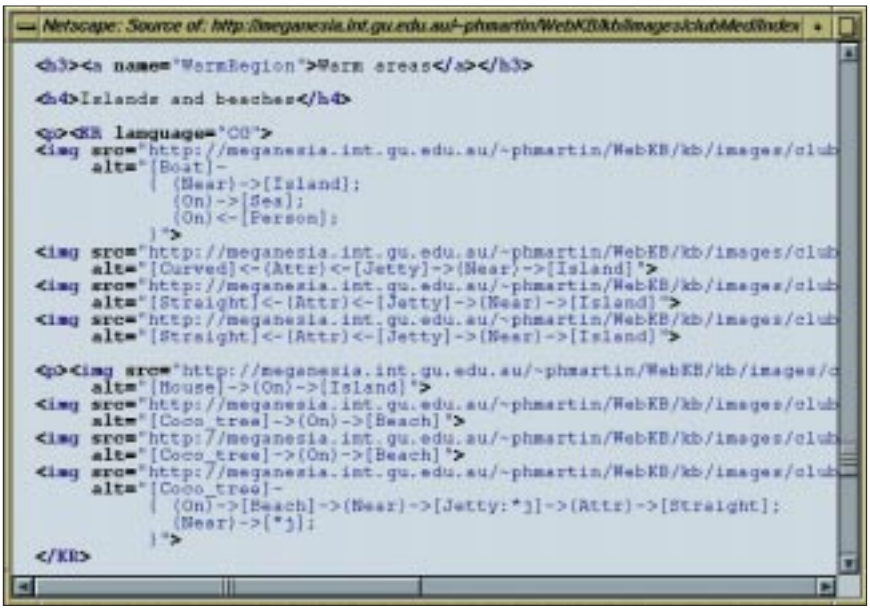

Figure 6. The HTML source code of the image indexation in Figure 5.

Indexing any document element using knowledge

A *document element* is any textual or HTML data, such as a sentence, section, or reference to an image or entire document. This definition excludes binary data but includes textual knowledge statements. WebKB lets users index any DE of a Web-accessible document (or later of our repository) with knowledge statements or connect DEs by relations. Figure 4 shows an example of each case.

XML provides more ways to isolate and reference DEs than HTML. Because WebKB exploits the capacities of Web browsers, the WebKB users can use the XML mechanisms. However, XML does not help users annotate others' documents, because DEs cannot be referenced if the documents' authors have not been explicitly delimited them. Therefore, the WebKB facility of referring to a DE by specifying its content and its occurrence number will still be useful.

A simple example. The indexation notations in Figure 4 let the statements and the indexed DEs be in different documents. Thus, any user can index any element of a document on the Web. Figure 1 presents a general interface for knowledge-based queries and shows how a document containing knowledge must be loaded in the WebKB processor before being queried.

WebKB also lets the author of a document index an image with a knowledge statement directly stored in the "`alt`" field of the HTML "`img`" tag used to specify the image. We use this special indexation case to present a simple illustration of WebKB's features. The example in Figure 5 is a good synthesis but does not represent the general use of WebKB, because it mixes the indexed source data (in this case, a collection of images), their indexation, and a customized interface to query them in a single document. Figure 6 shows a part of this document that illustrates the indexation. Figure 7 displays the results of the query in Figure 5.

Lexical and structural query commands. Because WebKB proposes knowledge representation, query commands, and a script language, we have not felt the need to give it a lexical and structural query language as precise as those in Harvest, WebSQL, and WebLog. Instead, we have implemented some Unix-like text processing commands to exploit Web-accessible documents or databases and generate other documents—for example, `cat`, `grep`, `fgrep`, `diff`, `head`, `tail`, `awk`, `cd`, `pwd`, `wc`, and `echo`. We added the hyperlink path exploring the command `accessibleDocFrom`. This command lists the documents directly and indirectly accessible from given documents within a maximal number of hyperlinks. For example, the following command lists the HTML documents accessible from www.foo.bar/foo.html (maximum two levels) and that include the string `knowledge` in their HTML source code:

```
accessibleDocFrom -maxlevel 2
  - HTMLonly http://www.foo.bar/
foo.html | grep knowledge
```

Knowledge query commands. WebKB has commands for displaying specializations,

Figure 7. The document generated in response to Figure 5's query.

generalizations of a concept or relation type, or an entire CG in a knowledge base. At present, queries for CG specializations only retrieve connected CGs: the processor cannot retrieve paths between concepts specified in a query. If a retrieved CG indexes a document element, we can present it instead of the CG. (Figure 7 gives an example.) In both cases, we generated hypertext links to reach the source of each answer in its original document—WebKB will actually present a slightly modified copy of this original document to instruct the Web browser to display and highlight the selected answer in its source document. What follows is an example of such an interaction, assuming that www.bar.com/example.html is the file where the indexation in Figure 4, and `Something` is the most general concept type.

```
> load http://www.bar.com/
    example.htm
> spec [Something]->)color->
    [color: red]
 [Color: red]<-(Color)<-
    [vehicle->(attr)->[damaged]
    Source
> use Repr //display represented
    Des
> spec [Something]->(Color)->
    [Color: red]
  the red damaged vehicle
    Source
```

Queries for specializations give users some freedom in the way they expresses queries; they can do searches at a general level and subsequently refine them according to the results. However, they must know the exact names of types. To improve this situation, WebKB lets users give only a substring of a type in a query CG if they prefixed this substring by the character `%`. WebKB generates the actual requests by replacing the substring with the manually and automatically declared types that include that substring. WebKB discards replacements that violate the constraints imposed by relation signatures or individual types. Then, it displays and executes each remaining request. For example, `spec [%thing]` will trigger the generation and execution of `spec [something]`.

Users can combine knowledge query commands with the script language to generate complex documents, perform consistency tests on the knowledge base, or solve problems procedurally. The WebKB site provides many examples of queries and scripts; one script solves the Sisyphus-I room allocation problem (meganesia.int.gu.edu.au/~phmartin/WebKB/kb/sisyphus1.html). You are invited to test these examples at meganesia.int.gu.edu.au/~phmartin/WebKB or www.int.gu.edu.au/phmartin/WebKB.

Knowledge generation commands. The only type of knowledge generation commands in WebKB are commands that join CGs. We can define various kinds of joins but WebKB only proposes joins that, given a set of CGs, create a new CG specializing each of the source CGs. Although we insert the result in the CG base, it might not represent anything true for the user but provides a device for accelerating knowledge representation. For instance, in WebKB, we can collect and automatically merge CGs related to a type with a command—for example, `spec [TypeX] | maxjoin`. The result can then serve as a basis for the user to create a type definition for TypeX.

The following is a concrete example for the maximal join command:

```
> maxjoin  [Cat]->(On)->[Mat]
     [Cat:Tom]->(Near)->[Table]
  [Cat:Tom]-  {   (On)->[Mat];
              }
```

A scalable cooperatively built knowledge repository

Ontology servers support shared knowledge repositories—for example, the Ontolingua ontology server and Ontosaurus. However, they are not usable for managing large quantities of knowledge, and apart form AI-Trader,[10] they do not allow the indexation and retrieval of parts of documents. Support of cooperation between the users is essen-

tially limited to consistency enforcement, annotations and structured dialogues, as in APECKS, Co_4, and Tadzebao. We are currently extending WebKB to handle a knowledge repository. (For more details, see meganesia.int.gu.edu.au/~phmartin/WebKB/doc/coopKBbuilding.html). However, we address scalability by

- implementing a knowledge-based system that reuses FastDB;
- using visualization techniques (mainly the handling of aliases for terms and the generation of views) that avoid lexical conflicts and let users focus on certain kinds of knowledge;
- using protocols that let users solve semantic conflicts by inserting new terms and relations in the common ontology and, in some cases, in the knowledge of other users; and
- using conventions for representing knowledge that improve the automatic comparison of knowledge from different users and hence their consistency and retrieval.

CURRENT INFORMATION retrieval techniques are not knowledge-enabled and hence cannot give precise answers to precise questions. To overcome this problem, a current trend on the Web is to let users annotate documents using metadata languages. WebKB lets its users combine lexical, structural, and knowledge-based techniques to exploit or generate Web documents. The scalable knowledge repository we are building will permit the fusion and reuse of knowledge from various sources. In an operational context, these knowledge-based features need to be combined with more traditional information retrieval ideas that give both coarse-grained search capabilities and the fine-grained, precision-based knowledge retrieval we describe here.

References

1. J.F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Mass., 1984.
2. P. Martin and P. Eklund, "WWW Indexation and Document Navigation Using Conceptual Structures," *Proc. 2nd IEEE Int'l Conf. Intelligent Processing Systems, ICIPS '98*, IEEE Press, Pistcataway. N.J., 1998, pp. 217–221.
3. P. Martin and P. Eklund, "Embedding Knowledge in Web Documents," *Proc. 8th Int'l World Wide Web Conf.*, Elsevier, 1999, pp. 324–341.
4. S. Decker et al., "Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information," *Semantic Issues in Multimedia Systems*, R. Meersman et al., eds., Kluwer Academic Publisher (in press), Boston, 1999.
5. O. Haemmerlé, *CoGITo: une plate-forme de developpement de logiciels sur les graphes conceptuels* (CoGITo: A Conceptual Graph Workbench), PhD thesis, Montpellier II Univ., France, Jan. 1995.
6. G. Ellis, *Managing Complex Objects*, PhD thesis, Dept. of Computer Science, Queensland University, Australia, 1995.
7. J. Nanard et al., "Integrating Knowledge-based Hypertext and Database for Task-Oriented Access to Documents," *Proc. DEXA'93, Lecture Notes in Computer Science*, Vol. 720, Springer-Verlag, New York, 1993, pp. 721–732.
8. P. Martin, "Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition," *Proc. 4th Peirce Workshop*, 1995. www.inria.fr/acacia/Publications/1995/peirce95phm.ps.Z (current May 2000).
9. P. Martin, *Exploitation de graphes conceptuels et de documents structure* (Exploitation of Conceptual Graphs and Structures Documents for Knowledge Acquisistion and Information Retrieval), PhD Thesis, University of Nice - Sophia Antipolis, France, 1996.
10. A. Puder and K. Romer, "Generic Trading Service in Telecommunication Platforms," *Proc. 5th Int'l Conf. Conceptual Structures, Lecture Notes in Artifical Intelligence*, Vol. 1257, Springer-Verlag, New York, 1997, pp. 551–565.

Philippe Martin is a research fellow at Griffith University's Gold Coast Campus, Australia. His main interests are knowledge representation, sharing, and retrieval. He has an engineering degree and a PhD in software engineering both from the University of Nice–Sophia Antipolis, France. Contact him at Griffith University, School of Information Technology, PMB 50 Gold Coast MC, QLD 9726, Australia; philippe.martin@gu.edu.au.

Peter W. Eklund is the Foundation Chair of Information Technology at Griffith University's Gold Coast Campus. He is also a key researcher at the Distributed Systems Technology Center in Brisbane. His main interests are knowledge ordering, visualization, and management. He graduated in mathematics from Wollongong University, Australia, and has an MPhil from Brighton University, UK, and a PhD in computer science from Linkoping University, Sweden. Contact him at Griffith University, School of Information Technology, PMB 50 Gold Coast MC, QLD 9726, Australia; p.eklund@gu.edu.au.