Knowledge Retrieval and the Word Wide Web

Philippe Martin and Peter Eklund

Griffith University, School of Information Technology, PMB 50 Gold Coast MC, QLD 9726 Australia
Tel: +61 7 5594 8271; Fax: +61 7 5594 8066; Email: {p.eklund,philippe.martin}@gu.edu.au

1 Introduction

Large-scale search engines for the WWW retrieve entire documents effectively. However, they can be considered imprecise because they do not exploit and hence retrieve the semantic content of Web documents. Such content cannot yet be automatically extracted from general documents. Manually structuring Web documents, e.g. via mark-up languages such as XML[1], allows more precise information to be retrieved using string-matching and structure-matching tools, e.g. Web robots such as Harvest[2], WebSQL[3] and WebLog[4]. However, this approach is not scalable because *fine-grained* information is only retrieved if the documents are thinly structured and the querier knows the structures, their exact names and forms. More flexible and precise knowledge representation and retrieval can be achieved with knowledge representation languages that support logic inference. Many "metadata" languages are currently being developed to allow people to index Web information resources by 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.

A first requirement for that is that the metadata language is sufficiently *intuitive and concise to be* **easy** *to use* by people (after a short period of training). Most current knowledge-oriented metadata languages are built above XML, e.g. RDF[5] and OML[6]. The choice of XML as an underlying format ensures that standard XML tools will be usable to exchange and parse these metadata languages. However, since XML is verbose, the metadata languages built above XML are also verbose and are difficult to use without specialized editors. Such editors do not eliminate the need for people to use a language for representing knowledge (except in application-dependent editors that only allow predefined "frames" to be filled). Consequently, as noted by the authors of Ontobroker[7] [1], with XML-based languages information has to be written in two versions, one for machines and another for humans. Additionally, standard XML tools are of little interest to manage these languages since specialized editors, analyzers and inference engines are required. To reduce information redundancy, Ontobroker provides a notation for embedding attribute-value pairs inside an HTML hyperlink tag. These tags may be used by the document's author to delimit an element to represent. Thus, each element may be implicitly referenced in the knowledge statement within the tag enclosing the element. When a final version of RDF is recommended by the Word Wide Web Consortium[8], a wrapper can be added to Ontobroker for automatically generating RDF definitions from Ontobroker metadata, thus making them accessible to a broader community.

Along this same line, a document's author should be allowed to let some *knowledge statements be visible* to the reader. This is an obvious requirement when an especially intuitive notation can be used, e.g. when graphics can be made with a visual language[9] or sentences can be written using a "controlled language"[10], — a subset of natural language which eliminates sources of 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, a knowledge base and its associated documentation can be integrated within the same document and both accessed using classic searches (string-matching, navigation from the table of content, etc.) as well as knowledge-based searches. Though the Ontobroker metadata language was designed to reduce information redundancy, statements cannot be shown since they are within HTML tags. Futhermore, like 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, since the indexation of document elements are made via HTML tags, only the document's author can index any of its parts. Others are limited to only those elements that are accessible via URLs.

---

[1] http://www.w3.org/XML/
[2] http://harvest.transarc.com/
[3] http://www.cs.toronto.edu/˜websql/
[4] http://www.cs.concordia.ca/˜special/bibdb/weblog.html
[5] http://www.w3.org/RDF/
[6] http://wave.eecs.wsu.edu/CKRMI/OML.html
[7] http://www.aifb.uni-karlsruhe.de/WBS/broker/
[8] http://www.w3.org/
[9] http://www.cpsc.ucalgary.ca/˜kremer/home.html#visualLanguages
[10] http://www-uilots.let.uu.nl/Controlled-languages/

A metadata language should also be sufficiently *precise and general* to allow users to 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 allowing the user to exploit the expressivity of the formal model. Any formalism equivalent to first-order logic and permitting the use of contexts is an appropriate candidate, e.g. KIF[11] and Conceptual Graphs (CGs)[12] [2]. It is important not to restrict users but, for efficiency reasons, a search engine may ignore some features in knowledge statements. For example, a CG-based search engine may 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 RDF are general but not precise in the sense that they are oriented towards the representation of entire documents (not arbitrary parts of them) and do not propose conventions to represent logic-based features, e.g. quantifiers and operators. This limits the capacity of the statements to be shared.

In summary, the three first requirements for *precise, flexible and scalable* information retrieval implies (i) several easy to use notations, some intuitive, some precise and expressive, and (ii) the possibilities to insert them anywhere in a Web document. We have satisfied these requirements by building a Web-accessible tool (CGI server[13]) named WebKB[14] [3][4] which interprets "chunks" of knowledge statements in Web documents. Each chunk, i.e. each group of statements, must be delimited by two special HTML marks ("`<KR>`" and "`</KR>`") or the strings "$(" and ")$". These chunks are visible unless the document's author hides them with HTML comment tags. The knowledge representation language used in each chunk must be specified at its beginning, e.g.: "`<KR language="CG">`". At present, WebKB can interpret the linear notation of CGs plus less expressive but simpler linear notations we have invented: a formalised English, a frame-like CG linear notation and structures that relate document elements by semantic relations (some of these structures come from HTML). These simpler notations are translated into CGs. This formalism has been chosen first because it has a graphical notation and a linear notation, both concise and easily comprehensible, and secondly because 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, statements and queries may be made at different levels of granularity. In the future, other notations may be accepted and other formalisms exploited.

Another requirement is that *not all the terms in the knowledge statements should have to be explicitly declared and organized by each user*. Indeed, declaring and organizing terms is a tedious and often complex work that detters most users, and 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 semi-formal knowledge representation languages such as concept maps[15], 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. We show below how the imprecision may partially be compensated by exploiting ontologies. Another informal feature accepted by WebKB are notations for sets within CGs: WebKB ignores them during searches but displays each retrieved CG in the form it was entered (thus, notations for sets may be seen).

HTML and XML do not allow a user to reference — and hence index — any part of a document that s/he has not created. An *indexation notation allowing a document element to be referred by its content or occurrence in a document* is required. WebKB provides such a notation.

Simply representing knowledge within documents is insufficient, *knowledge-based 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 the place of the commands. In the hypertext literature, such a technique is known as *dynamic linking*, and the generated document is called a *dynamic document* or a *virtual document* [7]. This idea has many applications, e.g. adapting a document content to a user. A procedural or declarative language is necessary to combine the commands and their results. Web robots (e.g. Harvest, WebSQL, WebLog) perform some document generation in that way but current metadata languages only allow knowledge representation. WebKB permits the generation of virtual documents and combines lexical, structural and knowledge-based data management by proposing (i) commands for searching and joining CGs, (ii) Unix-like file management commands working on Web-accessible documents, (iii) a simple Unix shell-like script language to combine commands. These commands may be inserted in documents. They may also be directly sent to WebKB by programs or manually from form-based interfaces, e.g. the WebKB interfaces. Figure 1 shows the WebKB tool menu and the "Knowledge-based Information Retrieval/Handling Tool".

The four following sections respectively illustrate the ideas of the last four paragraphs. Though 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 to exploit. To allow users to benefit from the knowledge of users they do not know, we are currently extending WebKB to handle *a cooperatively built knowledge repository*. We address this issue in section 6.

---

[11] http://logic.stanford.edu/kif/kif.html
[12] http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/CGs.html
[13] http://www.w3.org/CGI/
[14] http://meganesia.int.gu.edu.au/~phmartin/WebKB/
[15] http://www.cpsc.ucalgary.ca/~kremer/home.html#CM

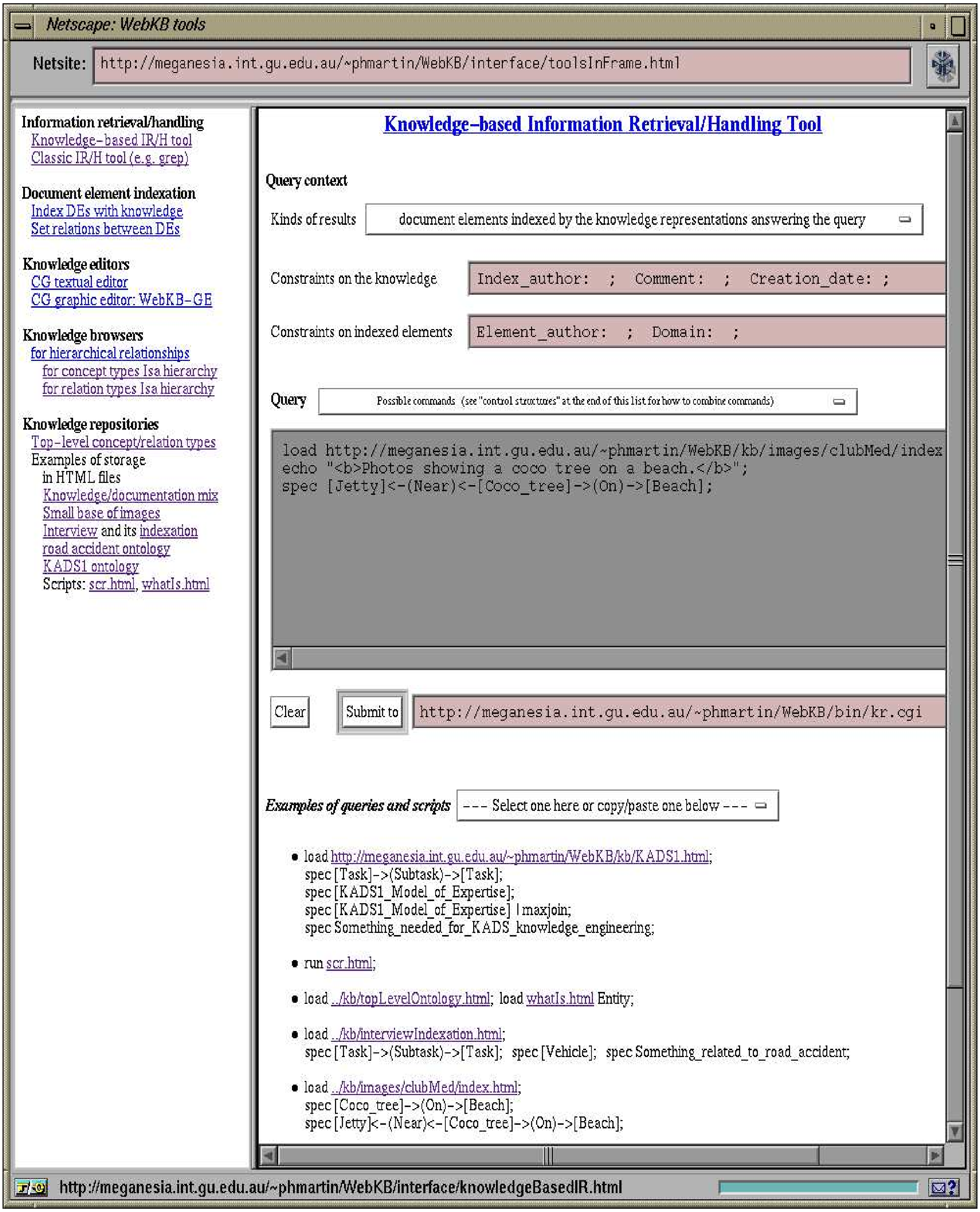

Figure 1: The WebKB tool menu and knowledge-based Information Retrieval/Handling Tool. *The example query shows how a document containing CGs (indexing images) is loaded into the WebKB processor and then how the command "spec" (which looks for specializations of a CG) can be used to retrieve CGs and the images they index. According to the value selected in the "kinds of results" option (cf. top right of the figure), the images, but not the knowledge statements, will be presented. A similar query and its results is shown in Figures 5 and 7.*

2 Representing Knowledge

2.1 Knowledge Representation Languages vs XML-based Metadata Languages

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

```
<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 KIF, CGs and RDF.

The CG representation (top) seems simpler than the others. The semantic network structure of CGs (i.e. concepts connected by relations) has three advantages: (i) it restricts the formulation of knowledge without compromising expressivity and this tends to ease knowledge comparison from a computational viewpoint; (ii) it encourages the users to explicit relations between concepts (as opposed, for instances, to languages where "slots" of frames or objects can be used); (iii) it permits a better visualization of relations between concepts.

2.2 Less Expressive but More Intuitive Notations

Even if CGs seem relatively intuitive, they are not readable by everyone. In restricted cases, simpler notations may be prefered. For instance, Figure 3 shows notations that are accepted by WebKB 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];
}
```

```
    /* Structured text (":" ends the name of a "typical" relation,
                        "=>" 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.

3 Allowing Undeclared Terms in Knowledge Statements

The user may not want to take the time to declare and order most of the terms s/he uses when representing knowledge. This may, for example, be the case when a user indexes sentences from various documents for private knowledge organisation purposes.

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

- If knowledge statements are made from concepts linked by basic relations, i.e. 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[16] [8] [9] collecting common thematic, mathematical, spatial, temporal, rhetorical and argumentative relations types.
- WebKB can use relation signatures to give suitable types to the undeclared terms used as concept types. For instance, in the top-level ontology proposed by WebKB, 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*.
- We have merged the natural language ontology WordNet[17] (120,000 words linked to 90,000 concept types) into our top-level ontology (cf. [8] [9]). When the WebKB shared repository is implemented and initialized with these ontologies, it will be possible for WebKB to semi-automatically 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. Consider for example, the following CG where the terms *Cat* and *Table* have not been declared: `[Cat]->(On)->[Table]`. In WordNet, the word *cat* has 5 meanings (feline, gossiper, X-ray, beat and vomit) and the word *table*, 5 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?".
- Knowledge statements are more readily comparable if they follow the same conventions. The convention of using basic relations is thus important. (The opposite convention — using primitive concepts and complex relations — would be much harder to follow). Consider for example the sentence "Mary is 20 years old". Following our conventions it is better to use the concept type *Age*, e.g. `[Person:"Mary"]->(Chrc)->[Age:@20]`,

[16] http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/topLevelOntology.html
[17] http://www.cogsci.princeton.edu/~wn/

rather than the relation type *Age*, e.g. `[Person:"Mary"]->(Age)->[Integer: 20]`, unless this relation type has been predefined by a user: [18]

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

By default, WebKB enforces declared terms in the CG linear notation but permits undeclared terms in the other (simpler) notations (cf. 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. Quoted sentences may also be used: they are understood by WebKB as individual concepts of 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 may combine HTML or XML features with knowledge statements, e.g. s/he may put some types in italics or make them the source of hypertext links.

4 Indexing any Document Element using Knowledge

4.1 General Cases

We call a Document Element (DE) any textual/HTML data, e.g. a sentence, a section, a reference to an image or to an entire document. This definition excludes binary data but includes textual knowledge statements. WebKB allows users to index any DE of a Web-accessible document (or later of our repository) by knowledge statements, or connect DEs by relations. Figure 4 shows an example of each case.

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

$(DEconnection
   (Context: Language: CG;
             Ontology: http://www.bar.com/topLevelOntology.html;
             Repr_author: phmartin; Creation_date: Mon Sep 14 02:53:36 1998;)
   (DE: {Document: http://www.bar.com/example.html} )
   (Relation: Summary)
   (DE: {Document: http//www.bar.com/example.html} {section title: Abstract})
)$
```

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

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

4.2 A Simple Example

The above indexation notations allow the statements and the indexed DEs to be in different documents. Thus, any user may 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.

[18] This solution implies that the inference engine expands the relation type definition when comparing graphs. Few CG engines can perform type expansion.

WebKB also allows *the author of a document* to index an image by a knowledge statement directly stored in the "alt" field of the HTML "img" tag used to specify the image. We use this special case of indexation to present a simple illustration of WebKB's features. This example, shown in Figure 5, is a good synthesis but in no way representative of the general use of WebKB — it is not representative 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. The result of the query shown in Figure 5 is displayed in Figure 7.

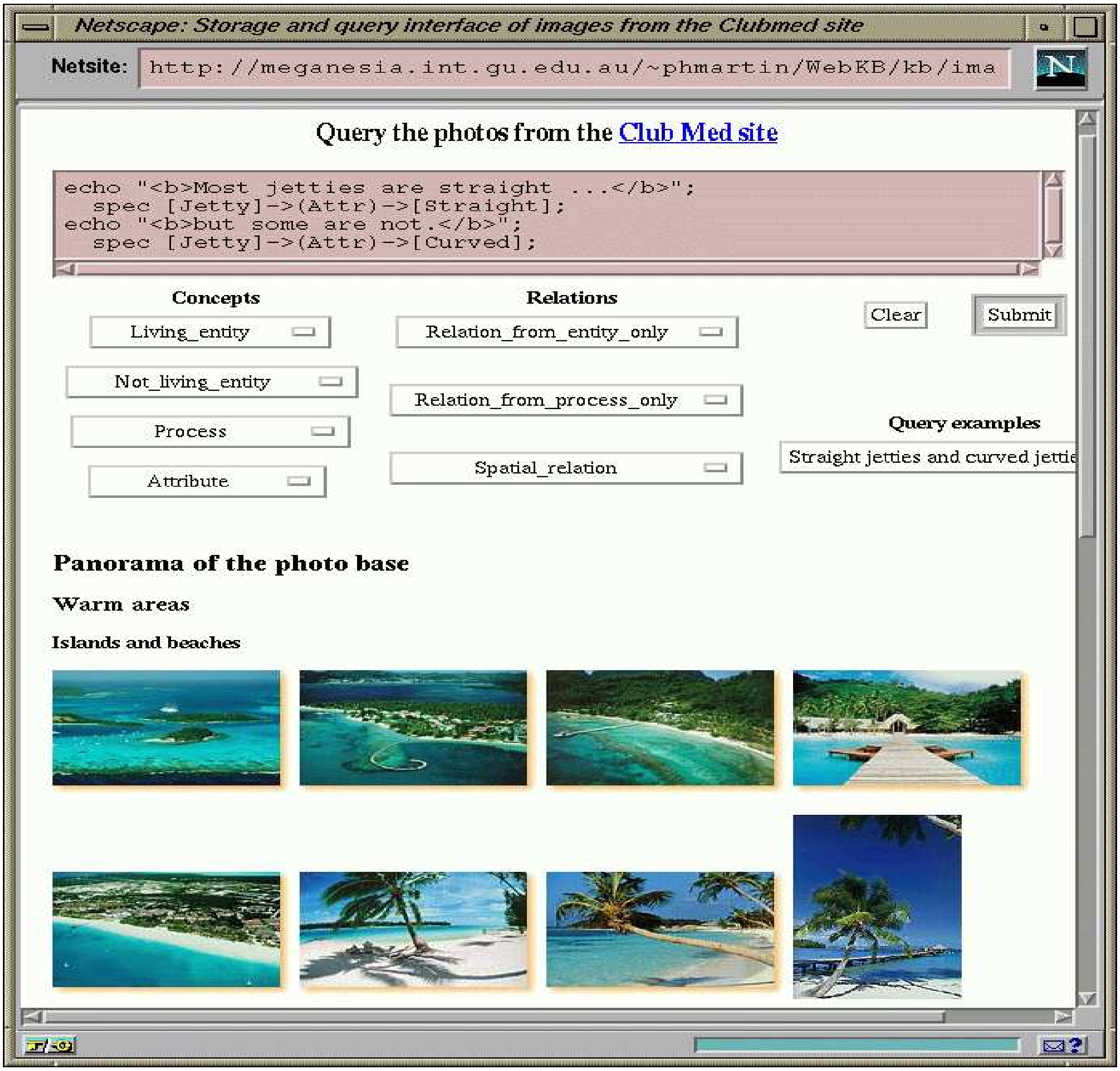

Figure 5: Images, knowledge indexations and a customized query interface contained within a same document *(the example query shows how the command "spec" which looks for specializations of a CG can be used to retrieve images indexed by CGs. The results are shown in Figure 7).*

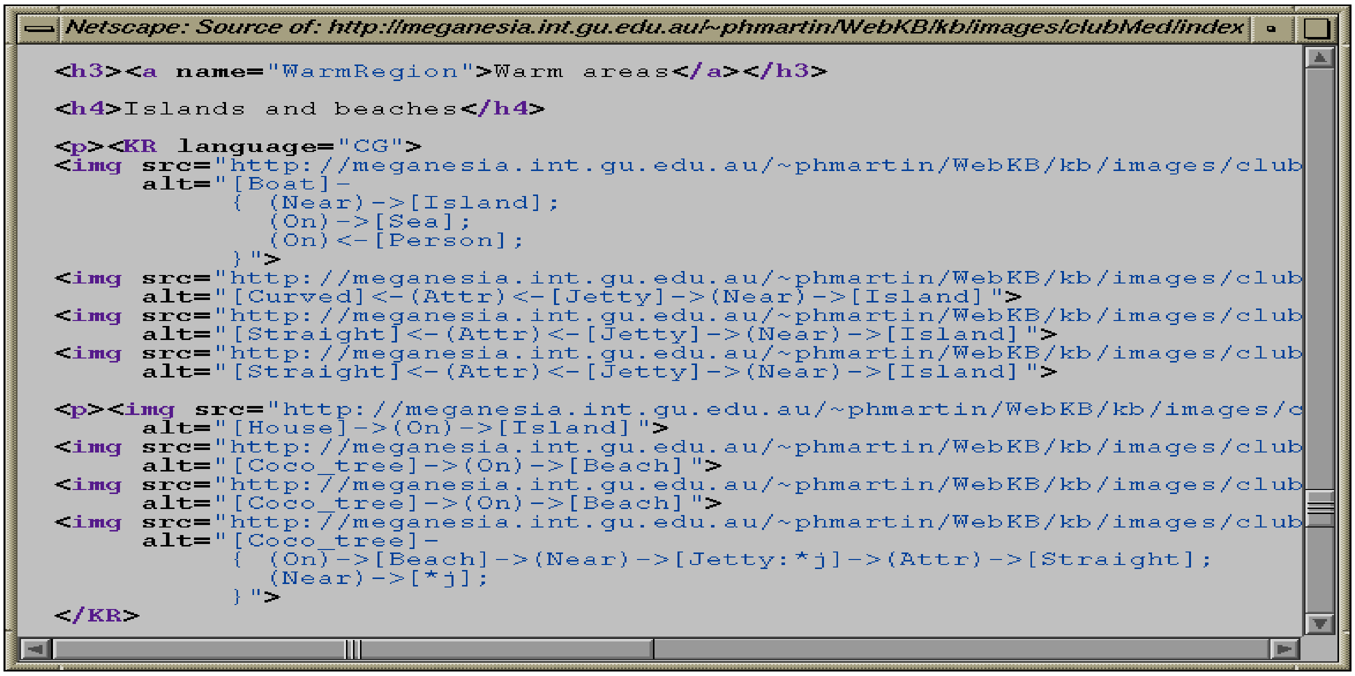

Figure 6: The HTML source of the indexation of the images shown in Figure 5.

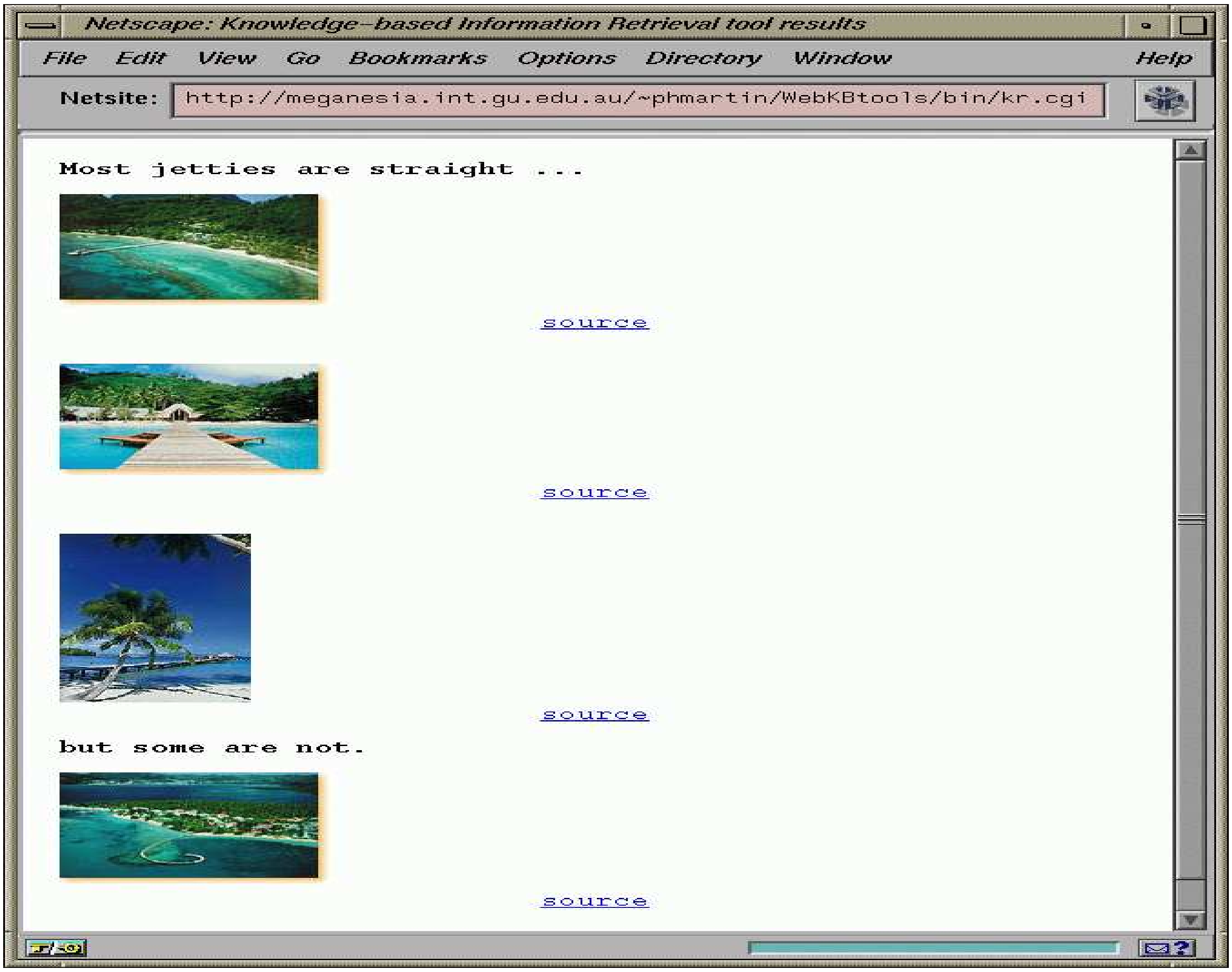

Figure 7: The document generated in response to the query in Figure 5.

5 Commands to Exploit Knowledge and Documents

5.1 Lexical and Structural Query Commands

Because WebKB proposes knowledge representation and 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 of Harvest, WebSQL and WebLog. Instead, we have implemented some Unix-like text processing commands for exploiting Web-accessible documents or databases and generating other documents, e.g. cat, grep, fgrep, diff, head, tail, awk, cd, pwd, wc and echo. We added the hyperlink path exploring 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 http://www.foo.bar/foo.html (maximum 2 levels) and that include the string "nowledge" in their HTML source code.

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

5.2 Knowledge Query Commands

WebKB has commands for displaying specializations or 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, it may be presented instead of the CG (Figure 7 gives an example). In both cases, hypertext links are generated to reach the source of each answer presented in its original document — a slightly modified copy of this original document will actually be presented by WebKB in order 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 http://www.bar.com/example.html is the file where the indexation in Figure 4 has been stored, and *Something* is the most general concept type.

```
> load http://www.bar.com/example.html
> 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 the user some freedom in the way s/he expresses queries: searches may be done at a general level and subsequently refined according to the results. However, the exact names of types must be known. To improve this situation, WebKB allows the user to give only a substring of a type in a query CG if s/he prefixed this substring by the character %. WebKB generates the actual request(s) by replacing the substring by the manually/automatically declared types that include that substring. Replacements that violate the constraints imposed by relation signatures or individual types are discarded. Then, each remaining request is displayed and executed. For example, `spec [%thing]` will trigger the generation and execution of `spec [Something]`.

Knowledge query commands may be combined 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. For example, one script solves the Sisyphus-I room allocation problem[19]. The reader is invited to test these examples[20].

5.3 Knowledge Generation Commands

The only type of knowledge generation commands in WebKB are commands that join CGs. Various kinds of joins may be defined but WebKB only proposes joins which, given a set of CGs, create a new CG specializing each of the source CGs. Though the result is inserted in the CG base, it may not represent anything true for the user, but provides a device for accelerating knowledge representation. For instance, in WebKB, CGs related to a type may be collected and automatically merged via a command such as this one: `spec [TypeX] | maxjoin`. The result may 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];
              (Near)->[Table];
            }
```

[19] http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/sisyphus1.html
[20] http://meganesia.int.gu.edu.au/~phmartin/WebKB/, or if this server is down, http://www.int.gu.edu.au/~phmartin/WebKB/

6 A Scalable Cooperatively Built Knowledge Repository

Some servers, called ontology servers, support shared knowledge repositories, e.g. the Ontolingua ontology server[21] and Ontosaurus[22]. However, they are not usable for managing large quantities of knowledge and, apart form AI-Trader [10][23], they do not allow the indexation and retrieval of parts of documents. Finally, support of cooperation between the users is essentially limited to consistency enforcement, annotations and structured dialogues, as in APECKS[24], Co4[25] and Tadzebao[26]. We are currently extending WebKB to handle a knowledge repository. As this implementation has just begun, we do not detail this extension[27]. However, here are the four points through which we address scalability: (i) the implementation of the knowledge-based system reuses Shore[28], a scalable multi-user persistent object repository and algorithms allowing the exploitation of large-scale dynamic taxonomies efficiently[29]; (ii) some visualisation techniques (mainly the handling of aliases for terms and the generation of views) avoid lexical conflicts and enable users to focus on certain kinds of knowledge; (iii) some protocols allow users to solve semantic conflicts via the insertion of new terms and relations in the common ontology and, in some cases, in the knowledge of other users; (iv) some conventions for representing knowledge to improve the automatic comparison of knowledge from different users and hence their consistency and retrieval.

7 Conclusion

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 allow users to annotate documents using metadata languages. On the basis of ease and representational completeness, we have argued for the use of general and intuitive knowledge representation languages such as CGs rather than the direct use of XML-based languages. To allow users to represent knowledge at the level of detail they require, we have proposed simple notations for restricted knowledge representation cases and a technique allowing users to leave knowledge terms undeclared. Our tool WebKB supports this approach and allows its users to combine lexical, structural and knowledge-based techniques to exploit or generate Web documents. It is currently extended to handle a scalable cooperatively built knowledge repository.

References

[1] S. Decker, et al., Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Semantic Issues in Multimedia Systems, Kluwer Academic Publisher (in press), Boston, 1999.

[2] J.F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.

[3] Ph. Martin, and P. Eklund, “WWW Indexation and Document Navigation Using Conceptual Structures,” Proc. ICIPS’98, 2nd IEEE International Conference on Intelligent Processing Systems, IEEE Press, 1998, pp. 217-221.

[4] Ph. Martin, and P. Eklund, “Embedding Knowledge in Web Documents,” Proc. WWW8, 8th International World Wide Web Conference (in press), Toronto, Canada, May 1999.

[5] O. Haemmerlé, *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels*, Ph.D thesis, Montpellier II University, France, Jan. 1995.

[6] G. Ellis, Managing Complex Objects, Ph.D thesis, Queensland University, Australia, 1995.

[7] J. Nanard, et al., “Integrating Knowledge-based Hypertext and Database for Task-oriented Access to Documents,” Proc. DEXA’93, Prague, Springer Verlag, LNCS Vol. 720, Prague, 1993, pp. 721-732.

[8] Ph. Martin, “Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition,” Proc. Peirce’95, 4th Peirce workshop, California, Aug. 18, 1995. http://www.inria.fr/acacia/Publications/1995/peirce95phm.ps.Z

[9] Ph. Martin, *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l’acquisition de connaissances et la recherche d’informations*, PhD Thesis, University of Nice - Sophia Antipolis, France, Oct. 1996.

[10] A. Puder, and K. Romer, “Generic Trading Service in Telecommunication Platforms,” Proc. ICCS’97, 5th International Conference on Conceptual Structures, Springer Verlag, LNAI 1257, Aug. 1997, pp. 551-565.

---

[21] http://WWW-KSL-SVC.stanford.edu:5915/
[22] http://www.isi.edu/isd/ontosaurus.html
[23] http://www.vsb.informatik.uni-frankfurt.de/projects/aitrader/intro.html
[24] http://www.psychology.nottingham.ac.uk/staff/Jenifer.Tennison/APECKS/
[25] http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html
[26] http://ksi.cpsc.ucalgary.ca:80/KAW/KAW98/domingue/
[27] for more details, see http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/coopKBbuilding.html
[28] http://www.cs.wisc.edu/shore/
[29] http://www.cs.sfu.ca/cs/people/GradStudents/fall/personal/index.html