Embedding Knowledge in Web Documents: CGs versus XML-based Metadata Languages

Philippe Martin and Peter Eklund

Griffith University, School of Information Technology,
PMB 50 Gold Coast MC, QLD 9726 Australia
{p.eklund,philippe.martin}@gu.edu.au

Abstract. The paper argues for the use of general and intuitive knowledge representation languages for indexing the content of Web documents and representing knowledge within them. We believe these languages have advantages over metadata languages based on the Extensible Mark-up Language (XML). Indeed, the representation and retrieval of precise information is better supported by languages designed to represent semantic content and support logical inference, and the readability of such a language eases its exploitation, presentation and direct insertion within a document.To further ease the representation process, we propose techniques allowing users to leave some knowledge terms undeclared. We illustrate these ideas with WebKB[1], a precision-oriented information retrieval/annotation tool, and show how lexical, structural and knowledge-based techniques may be combined to retrieve or generate knowledge or Web documents. Finally, to overcome the scalability problems of storing knowledge within Web documents, we propose some ideas for scalable and cooperatively built knowledge repositories.

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[2], allows more precise information to be retrieved using string-matching and structure-matching tools, e.g. Web robots such as Harvest[3], WebSQL[4] and WebLog[5]. 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

[1] http://meganesia.int.gu.edu.au/~phmartin/WebKB/
[2] http://www.w3.org/XML/
[3] http://harvest.transarc.com/
[4] http://www.cs.toronto.edu/~websql/
[5] http://www.cs.concordia.ca/~special/bibdb/weblog.html

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[6] and OML[7]. 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[8] [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.

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.

---

[6] http://www.w3.org/RDF/

[7] http://wave.eecs.wsu.edu/CKRMI/OML.html

[8] http://www.aifb.uni-karlsruhe.de/WBS/broker/

[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] [10]. 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] [6][7] 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 [3] and Peirce [2]) 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

---
[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/

been knowledge-based (MacWeb [8] 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 are displayed).

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.

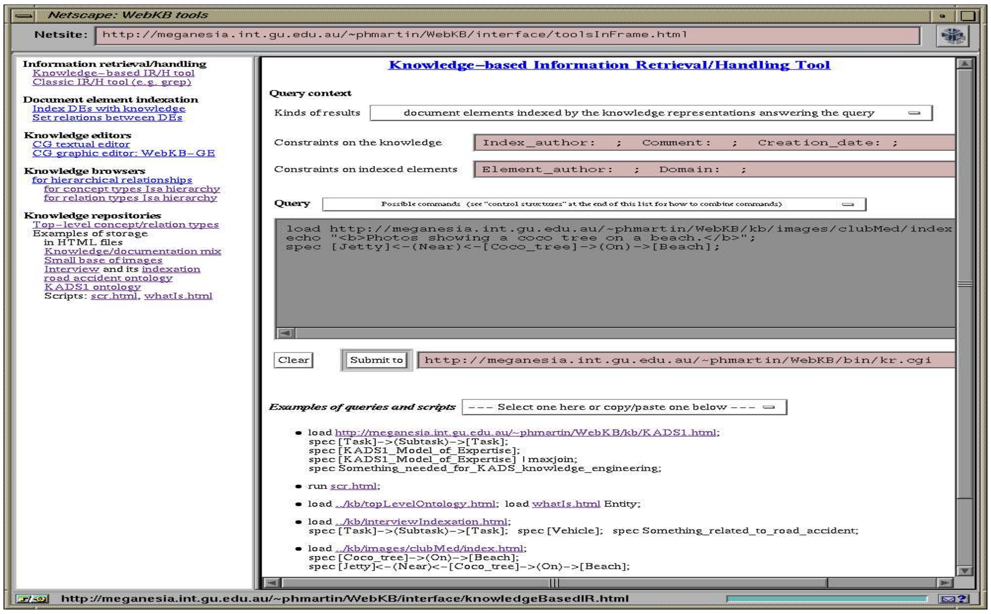


Fig. 1. The WebKB tool menu and knowledge-based Information Retrieval/Handling Tool. *This example shows how a document containing CGs is sent to the WebKB server and how the command "spec" is used to retrieve CGs and the images they index*

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

[15] http://www.cpsc.ucalgary.ca/˜kremer/home.html#CM

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* [8]. 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. Considering its features, WebKB may be seen as a knowledge-based *directed* Web robot and *private* annotation tool. To allow users to benefit from the knowledge of users they do not know, and therefore to enable WebKB to also be a knowledge-based *shared* annotation tool, we are extending it to handle *a cooperatively built knowledge repository.* We address this issue in section 6.

2 Representing Knowledge

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".

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.

Even if CGs seem relatively intuitive, they are not readable by everyone. Often, simpler notations could be used. For instance, WebKB accepts alternative notations for CGs. We call two of them "Frame-CGs" and "Formalised English". In Frame-CGs, the above sentence could be represented in that way: `[Mary. Age: a. Cousin: [Person. Age: a] ].` Here are two possibilities

```
<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>
```

Fig. 2. Comparing knowledge representation with CGs, KIF and RDF

in Formalised English: `John believes {Mary has for age A and has for cousin a person who has for age A}.` and `{Mary has for cousin a person who has for chrc an age chrc of Mary}(Believer:John).`

By default, WebKB accepts that statements expressed in Frame-CGs and Formalised English include undeclared terms. On the opposite, unless forewarned by the command "no decl", WebKB requires that terms used in CGs are declared.

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] [4] [5] 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. [4] [5]). 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*,

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

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]`,
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];
                              }
```

The commands "decl" and "no decl" enable the user to overide the default modes for the acceptatance of undeclared terms. Furthermore, 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 3 shows an example of each case.

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

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

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

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

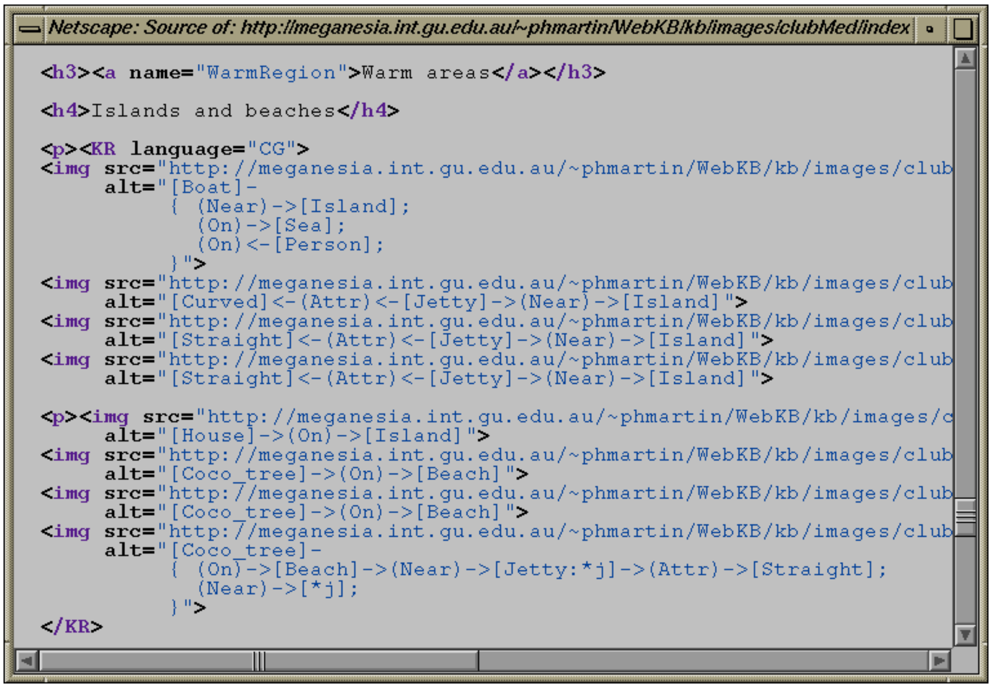
Fig. 3. A language for indexing or connecting any Web-accessible document element

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.

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 4 shows a part of this document that illustrates the indexation. The result of the query shown in Figure 5 is displayed in Figure 6.



Netscape: Source of: http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/clubMed/index

```
<h3><a name="WarmRegion">Warm areas</a></h3>

<h4>Islands and beaches</h4>

<p><KR language="CG">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Boat]-
          {  (Near)->[Island];
             (On)->[Sea];
             (On)<-[Person];
          }">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Curved]<-(Attr)<-[Jetty]->(Near)->[Island]">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Straight]<-(Attr)<-[Jetty]->(Near)->[Island]">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Straight]<-(Attr)<-[Jetty]->(Near)->[Island]">

<p><img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/c
     alt="[House]->(On)->[Island]">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Coco_tree]->(On)->[Beach]">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Coco_tree]->(On)->[Beach]">
<img src="http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/images/club
     alt="[Coco_tree]-
          {  (On)->[Beach]->(Near)->[Jetty:*j]->(Attr)->[Straight];
             (Near)->[*j];
          }">
</KR>
```

Fig. 4. The HTML source of the indexation of the images shown in Figure 5

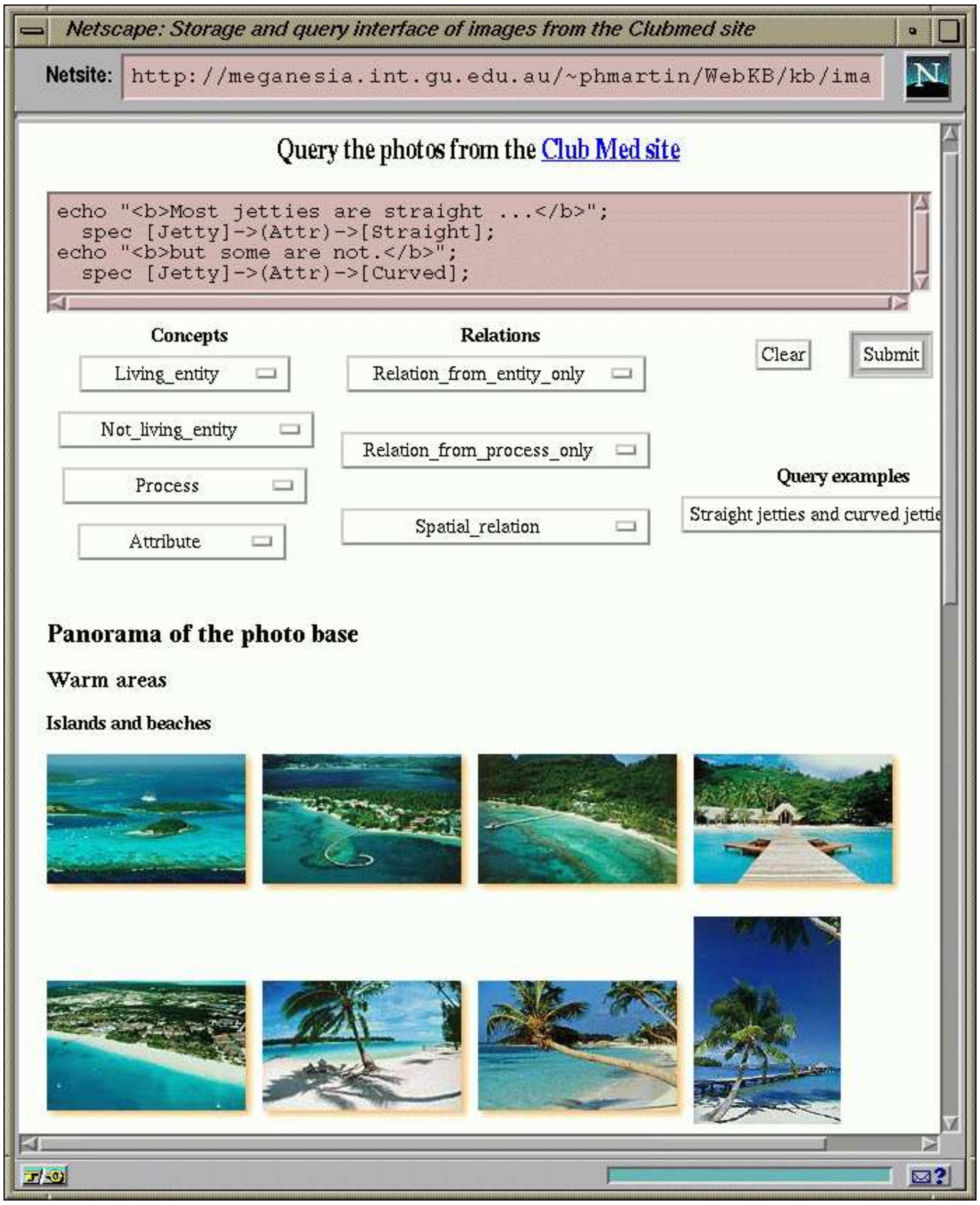


Fig. 5. Images, knowledge indexations and a customized query interface contained within a same document *(the example query shows how the command "spec" which looks for specializations of a CG can be used to retrieve images indexed by CGs. The results are shown in Figure 6)*

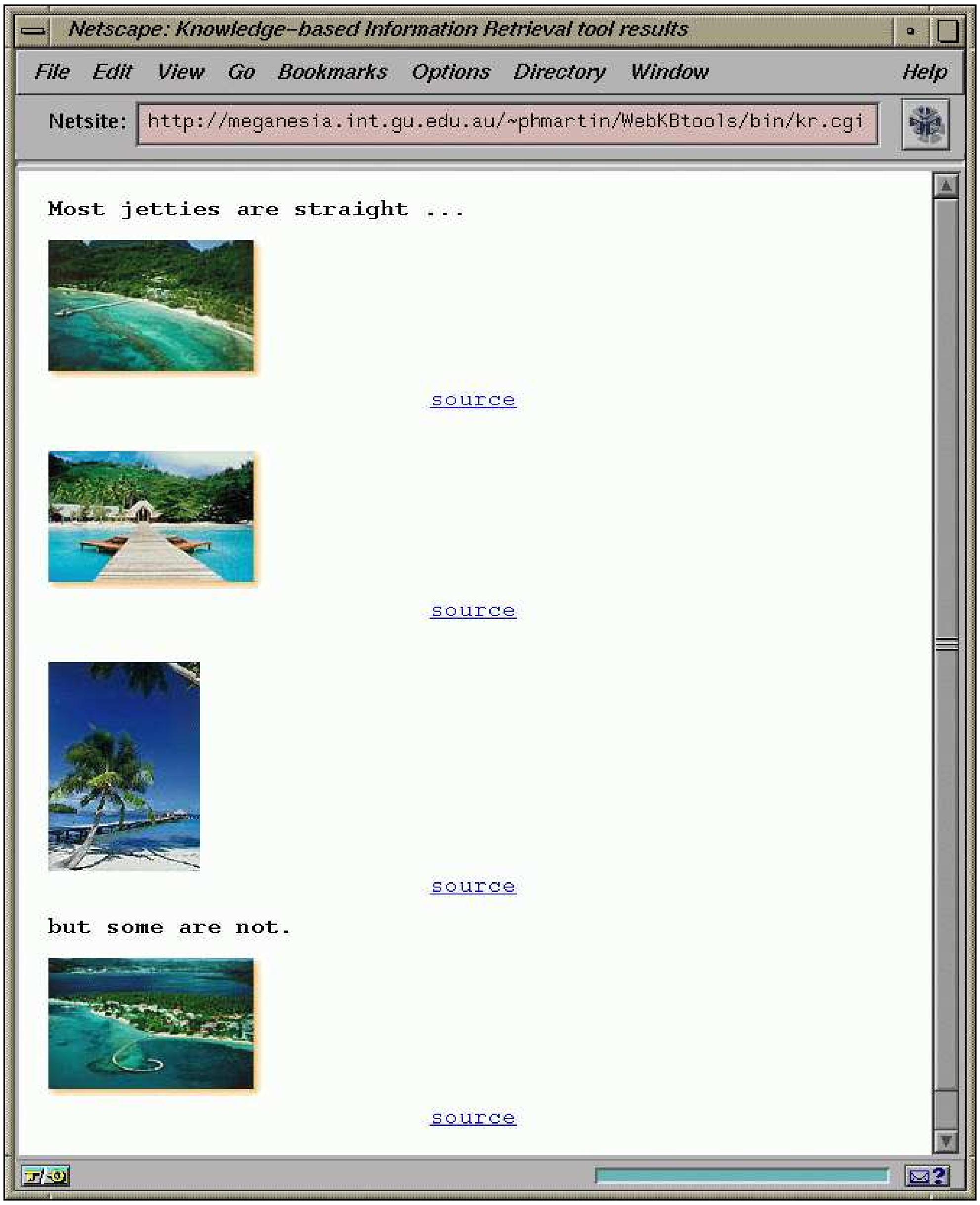


Fig. 6. The document generated in response to the query in Figure 5

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 word “knowledge” in their HTML source code.

```
> accessibleDocFrom -maxlevel 2 -HTMLonly http://www.foo.bar/foo.html
    | grep -i knowledge     //''-i'' to search without regard to case
```

Lexical/structural queries – and knowledge queries – may be embedded inside documents so parts of these documents may be generated by WebKB using document elements or knowledge stored in other documents. Alternatively, with HTML documents, Javascript may be used for associating a query to an hypertext link in such a way that the query is sent to the WebKB processor when the link is activated (then, as for any other query, the WebKB processor generates an HTML document that includes the results; if the query has been sent from a Web-browser, this “virtual” document is automatically displayed).

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 6 gives an example). In both cases, hypertext links are generated to reach the source of each answer presented in its original 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 3 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].

Here is an example of a script that shows that the procedural language frees us to add some special operators to our query language, such as the modal operators "few" and "most", since they are easily definable by the user.

```
spec [Something] | nbArguments | set nbCGs;
spec [Cat] | nbArguments | set nbCGsAboutCat;
set nbCGsdiv2 ‘expr $nbCGs / 2‘;
if ($nbCGsAboutCat > $nbCGsdiv2)
{ echo "Most CGs of the base are about cats"; }
```

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 Scalable Cooperatively Built Knowledge Repositories

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 from AI-Trader [9][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 five points through which we address scalability: (i) a scalable multi-user persistent object repository to support the storage and exploitation of knowledge structures (we have chosen Shore[28]); (ii) algorithms allowing the exploitation of large-scale dynamic taxonomies efficiently (we have chosen Fall's algorithms[29]; (iii) visualisation techniques (mainly the handling of aliases for terms and the generation of views) to avoid lexical conflicts and enable users to focus on certain kinds of knowledge; (iv) protocols to 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; (v) conventions for representing knowledge to improve the automatic comparison of knowledge from different users and hence their consistency and retrieval.

Though these five points permit the exploitation of a large knowledge repository (that is essential for efficiency reasons and practical use), it is also clear that for efficiency and reliability reasons, a unique server cannot be used to handle a universal knowledge repository by all Web users. Knowledge has to be distributed and mirrored on various knowledge servers. However, since there is no static conceptual schemas in knowledge bases, the techniques of distributed database systems - such as AlephWeb[30], Hermes[31] and Infomaster[32], cannot all be reused since they exploit a fixed conceptual schema (ontology) associated to each database. In knowledge bases, the ontology is constantly modified by the users.

---

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 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
30 http://www.pangea.org/alephweb.aleph/paper.html
31 http://www.cs.umd.edu/projects/hermes/
32 http://infomaster.stanford.edu/infomaster-info.html

A first step to the distribution of a knowledge repository is to duplicate it on several servers, with updates made on a server automatically duplicated in other servers. Some servers may be dedicated to searches and others to updates.

A second step is to have general servers and specialized servers. General servers would probably have knowledge bases with a content similar to the CYC knowledge base[33]. A specialised server would store the same knowledge as general servers plus knowledge related to a well-defined set of objects, e.g. knowledge expressed with the subtypes of certain types. Since these sets of objects are well-defined (extensively or via definitions), a general server would store the URLs of these servers and, when answering a query, would delegate the query to the relevant servers if more precision is required. These sets of objects might be determined by the managers of specialized servers, or according to the frequency of accesses to objects in knowledge repositories. Whatever the specialised server a user updates, if the knowledge it enters is relevant to other servers (e.g. if the knowledge is expressed with general terms), it should be automatically duplicated in these servers The rationale of all these duplications is to speed searches and simplify the query mechanisms by avoiding, whenever possible, parallel searches in various servers and then the composition of the results.

Other steps may be necessary, but what should be avoided in this knowledge-based approach (hence precision-oriented) is to let the specialized servers developp independently of each others instead of being part of a unique consistent virtual knowledge repository. Otherwise, conceptual queries and cooperation across the repositories would no longer be possible; this is the case in current traders where the repository most relevant repository to answer a query is automatically "guessed".

Finally, knowledge servers should not be limited to the storage of knowledge statements but should also allow the storage and handling of knowledge-based and document-based commands similar to the storage and handling procedure we described for documents.

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 knowledge representation/retrieval tool WebKB supports this approach and allows its users to combine lexical, structural and knowledge-based techniques to exploit or generate Web documents.

[33] http://www.cyc.com/

We have shown why the WebKB features are needed for precision and scalability. To overcome the scalabilitity limitations inherent to directed Web robots and private annotation tools, we are extending it to also handle a scalable cooperatively built knowledge repository.

Acknowledgments

This work is supported by a research grant from the Australian Defense, Science and Technology Organisation.

References

1. Decker, S., Fensel, D.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: Meersman, R. (eds.), Semantic Issues in Multimedia Systems, Kluwer Academic Publisher (in press), Boston (1999)
2. Ellis, G.: Managing Complex Objects. Ph.D thesis, Queensland University, Australia (1995)
3. Haemmerlé, O.: *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels*. Ph.D thesis, Montpellier II University, France (1995)
4. Martin, Ph.: Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition. In: Peirce'95, 4th Peirce workshop, California (1995) http://www.inria.fr/acacia/Publications/1995/peirce95phm.ps.Z
5. Martin, Ph.: *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 (1996)
6. Martin, Ph., Eklund, P.: WWW Indexation and Document Navigation Using Conceptual Structures. In: ICIPS'98, 2nd IEEE International Conference on Intelligent Processing Systems, IEEE Press (1998) 217–221
7. Martin, Ph., Eklund, P.: Embedding Knowledge in Web Documents. In: WWW8, 8th International World Wide Web Conference (in press), Toronto, Canada (1999)
8. Nanard, J., Nanard, M., Massotte, A-M., Djemaa, A., Joubert, A., Betaille, H., Chauch, J.: Integrating Knowledge-based Hypertext and Database for Task-oriented Access to Documents. In: DEXA'93, LNCS Vol. 720, Springer-Verlag, Prague (1993) 721–732
9. Puder, A., Romer, K.: Generic Trading Service in Telecommunication Platforms. In: ICCS'97, 5th International Conference on Conceptual Structures, LNAI 1257 Springer Verlag (1997) 551–565.
10. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA (1984)