WWW Indexation and Document Navigation using Conceptual Structures

Peter W. Eklund and Philippe Martin
School of Information Technology, Griffith University
Parklands Drive, Southport QLD 4215, AUSTRALIA
Email: `p.eklund@gu.edu.au`,
`philippe.martin@gu.edu.au`

Abstract — **WebKB is a public domain knowledge annotation toolkit allowing indices of any Document Element (DE) on the WWW to be built using annotations in a Knowledge Representation (KR) language: Conceptual Graphs. The language permits the semantic content and relationships to other DEs to be described precisely. Search can be initiated remotely, via a WWW-browser and/or other software. WebKB enables the document generation using inferences within the knowledge engine to assemble DEs. Additionally, the knowledge base provides an alternate index through which both query and direct hyper-link navigation can occur. This paper describes some of the key features of the toolkit and its approach to knowledge indexation.**

I. INTRODUCTION

WebKB [13] is a public domain knowledge annotation toolkit sharing many design principles with WWW-based public annotation tools, e.g. ComMentor [18] and HyperNews [9], and WWW-based traders, e.g. AlephWeb [17], NetRepository [10] and AI-trader [16]. Despite similarities, WebKB is intended to index DEs using a knowledge representation language and exploits the annotation to achieve information retrieval outcomes. This combination of document indexation with knowledge representation languages is called *knowledge indexation*.

HyperNews, AlephWeb and AI-trader allow indices to be created on entire documents only, WebKB allows indices on arbitrary parts of documents. AI-trader uses Conceptual Graphs [19] for indexing documents, while NetRepository uses Knowledge Interchange Format (KIF) [3, 4] for communication between knowledge servers. None of these tools, except WebKB, can generate documents as answers to user queries by assembling DEs via an inference engine.

WebKB is an extension of previous knowledge annotation efforts in a system called CGKAT [12]. Like CGKAT, WebKB allows indexing and retrieval over DE collections using a knowledge mark up language (KR language) either embedded in documents or stored in WWW-accessible databases. WebKB represents a substantial rewrite of CGKAT as an open web-accessible client/server architecture knowledge annotation system exploiting Java and Javascript enabled web browsers and incorporating two knowledge engines, CoGITo [6] and Peirce [2].

This paper is structured in five sections. First, we introduce the knowledge representation (KR) language used for indexing and retrieving DEs. We show how this KR can be used to reason about DE collections, how inference occurs, and how we can use the KR as the basis for knowledge retrieval. Second, we discuss the architecture of the toolkit. This architecture is independent of the KR language and all components can be inter-changed within the architecture. The issue of a shared vocabulary for knowledge annotation is raised in Section IV. In the last section, we provide a walk-through some of the system features and functionality and demonstrate how WebKB can be used to generate new documents through the assembly of DEs indexed by knowledge representation.

II. KNOWLEDGE REPRESENTATION

WebKB uses a knowledge representation called Conceptual Graphs [19] as an annotation language. Conceptual Graphs (CGs) are labeled multi-graphs: *concept* nodes are connected by *relation* nodes. *Contexts* can be introduced, for example to provide nested negation, and *co-referent links* added to a graph to express variables and their scope. In *simple CGs*, each node label is composed of a type and a generic/individual referent. A simple CG may be translated into an existential conjunctive and positive formula of first order logic. Fig. 1 is an example annotation of a source text *"John is repairing his own car."* and is in the CG linear form. In knowledge representation terms the "meaning" of the CG can be interpreted as the first-order logic formula as shown in Fig. 1.

```
[Repair]-
  {->(Agent)->[Person: John];
   ->(Object)->[Car]->(Owner)->[Person:John];
  }.
```

$$\exists x, \exists y, Repair(x) \wedge Person(John) \wedge Car(y) \wedge$$
$$Agent(x, John) \wedge Object(x, y) \wedge Owner(y, John).$$

Figure 1: A simple source text and its representation as a CG.

The vocabulary of concepts (like `Person` or `Car`) and relations (such as `Owner` and `Agent`) derive from a pre-defined ontology of types. An *ontology* refers to a dictionary of terms that may be used as the vocabulary for knowledge modeling. Such an ontology may be organized hierarchically.

The two sets of usable concept and relation types may be ordered in a partial order (subsumption relation), implying a specialization relation between CGs and in turn between DEs the CGs index. For example, Fig. 2 is a specialization of the graphs,

```
[Person:John]<-(Owner)<-[Car].

[Repair]-
         {<-(Object)->[Car];
          <-(Agent)->[Person:John];
         }.
```

```
 (Context:
  Language: CG;
  Element_author:
     Childrens book;
     Domain: repairing things;
  Repr_author:Peter;
     Comment:Simple example;
     Creation_date);
  (DE: John is repairing his own car.)
  (Repr: [Repairing]-
   {->(Agent)->[Man: John];
    ->(Object)->[Vehicle]
      <-(possession)->[Man: John];))$
```

Figure 2: An annotation contains information about the context (repairing things), the source of the text (Children's book), a date-stamp and the name of the annotator (Peter).

```
$(Indexation
 (Context:  Language: CG;
  Element_author: Peter.Eklund;
  Repr_author:Peter.Eklund;
  Creation_date: Mon Jun 01 22:13:29 1998;
  ) (DE: John is repairing his car.)
  (Repr: [Repairing]-
    { ->(Agent)->[Man: John];
      ->(Object)->[Vehicle]->(Owner)->[Man: John];
    }
  )
)$

$(Indexation
 (Context: Language: CG;
  Element_author: Peter.Eklund;  Domain:;
  Repr_author:Peter.Eklund;
  Creation_date: Mon Jun 01 22:18:54 1998;
 )
 (DE: Repairing involves a physical entity.)
 (Repr:[Repairing]->(Object)
       ->[Physical_entity].))$
```

Figure 3: These two CGs (resp. the DEs they index) generalize the CG (rep. the indexed DE) in Fig. 1. Thus, any of these CGs may be used to retrieve the CG or the DE in Fig. 1.

More expressive CGs may be built using more labels in the nodes, e.g. representing sets, context, and uncertainty but *complex CGs* cannot be compared using the specialization relation between simple CGs. Other matching or ordering relations need to be defined. Fig. 3 is an example CG which may be built to represent the sentence *"John has not repaired any cars today"*. WebKB can store and retrieve simple CGs but also contextualized (i.e. embedded) simple CGs: the query has to be a simple CG but when a simple CG is retrieved, it is presented with its contextualisation. For example, the CG in Fig. 4 would be retrieved with any of the preceedings graphs.

Definitions and Joins with Conceptual Graphs

WebKB accepts type definitions with necessary and/or sufficient or typical conditions. For example, the sentences *"A car mechanic is someone who often repairs cars"* and *"Repairing a car is dirty work and involves a spanner and a jack"* could respectively be represented in Fig. 5.

```
   [Time:today]<-(PTIM)<-[Proposition: p13
      [Repairing]-
         {->(Agent)->[Man: John];
          ->(Object)->[Vehicle]; }]  ].
```

Figure 4: An embedded CG representing the fact that "John is not repairing a vehicle today".

```
NSC for Car-mechanic (x) are
  [Repairing]-
  { ->(Agent)->[Man:*x];
    ->(Object)->[Vehicle];
    ->(Frequency)->[Frequency: often];}.
TC for Repairing (x) is
  [Repairing:*x]-
  {  <-(Succ)<-[Situation]->(Descr)
              ->[Proposition: p13
      (Not)<-[Proposition: p14 [Car:*c]
              ->(Attr)->[Functioning]];
     ->(Succ)->[Situation]->(Descr)
     ->[Proposition: p15
           [Car:*c]->(Attr)->[Functionning]
           [Person:*p]->(Attr)->[Dirty]];
     ->(Agent)->[Person:*p];
     ->(Object)->[Vehicle:*c];
     ->(Instrument)->[Spanner];
     ->(Instrument)->[Jack];   }.
```

Figure 5: Definition of the type Car-mechanic with Necessary and Sufficient Conditions, Definition of the type Repairing with typical conditions.

The graph-based structure of CGs recommends them for Information Retrieval (IR) and data discovery. For example, when a part of a CG specializes another CG, the two CGs can be joined to form a derived CG specializing both. There are many ways to define graph merging. For example, we define a *maximal join* as a join which "maximises" the number of matched nodes and then select a derived CG with a "minimal" number of concept and relation nodes. WebKB provide a command for such a join. For example, from the CGs
`[Person:John]<-(Owner)<-[Car]` and
`[Car]<-(Object)<-[Repair]->(Agent)->[Man:John]`,
our maximal join would produce the CG used in Fig. 2.

Figs. 6–9 show the process through which general CGs can be restricted and joined to form the required sentence for the `Repair` graph of Fig. 1. In Fig. 6, the concept type `Goal_directed_agent` is specialized to `Man`. Likewise, the concept type `Process` is specialized to `Repairing` and `UNIVERSAL` to `Vehicle`. In Fig. 9, the three graphs are joined on the `Man` and `Vehicle` concepts.

Figs. 6–9 show the process through which general CGs can be restricted and joined to form the required sentence for the `Repair` graph of Fig. 1. In Fig. 6, the concept type `Goal_directed_agent` is specialized to `Man`. Likewise,

```
[Repairing]-
   {->(Agent)->[Man: John];
    ->(Object)->[Vehicle]<-(Owner)
                         <-[Man: John];}.
```

Figure 6: An example of the maximal join of two conceptual graphs referred to in the text.

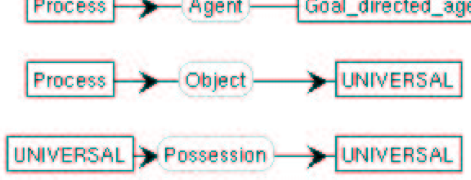

Figure 7: Graphs are posted to a "sheet of assertion" directly from the conceptual cannon.

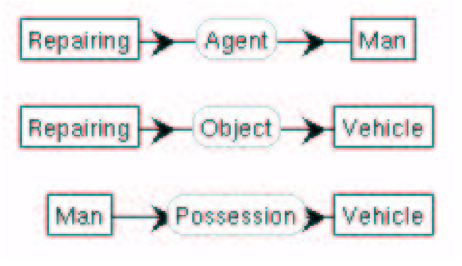

Figure 8: The general concepts are restricted to the types `Man`, `Vehicle` and `Repairing`.

the concept type `Process` is specialized to `Repairing` and `UNIVERSAL` to `Vehicle`. In Fig. 9, the three graphs are joined on the `Man` and `Vehicle` concepts.

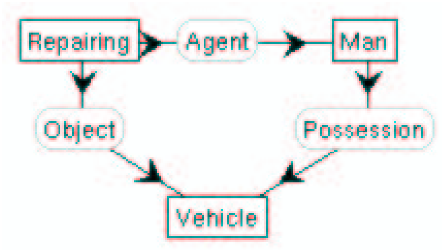

Figure 9: Where concepts match, the graphs can be joined. WebKB-GE [15] is a Java application that permits these graphical operations on conceptual graphs. These screen shots derive from WebKB-GE.

Given the open philosophy of WebKB, other knowledge representation languages and correspondent inference systems can be used, e.g. KIF [3, 4] or RDF (see `http://www.w3.org/RDF/`). The first selectable menu of WebKB allows the user to select the relevant knowledge engine. Note that in the annotations of Figs. 2 and 3 that the KR language is defined as "CG". This is a flag to alert the inference engine of the form of knowledge language.

Examples of Applications

Users of relational database query languages must understand the database structure, i.e. the exact names of the tables and their relations. The CG formalism can be used to exploit "search by content". For example, the specialization relation between CGs allows them, and the DEs they index, to be retrieved by specifying parts of their content with any supertype of the types they use. Additionally, if the calculation of the specialization relation takes into account type definitions, CGs may be retrieved even if their structure differs from the query structure — provided there exists a specialization relation between the panded.

When a DE *d* is composed of other DEs, some of which have been represented by CGs, these CGs may be merged using a maximal join to produce a default representation for the DE *d*. This technique is a key component of the method used by the information retrieval system RIME [7]. This technique can be used by the WebKB users for easing their representation of the content of a document.

Knowledge retrieval and merging may also guide knowledge modeling and DE indexation. As an example in the legal domain, CGs including concepts of types (or specializing types) "Legally-separated" or "Bankrupt-parterships" could be automatically retrieved and merged. The result may then serve as a guide to define the concept type "bankrupt partnerships between legally separated individuals". The CGs specializing this type definition could be automatically updated to use this new type. If these CGs indexed DEs, these DEs are now also accessible via queries using this new type.

When no results can be found for a query — there is no exact match — queries could be generalized to provide approximate results, e.g. by using only a part of the query or by using CGs which are supertypes of the query. This technique is used in ROCK [1] but not in WebKB.

III. ARCHITECTURE

An important element of our work is an open architecture. Separate, combinable and WWW-accessible tools permit ease of use and exchange with other tools thought more interesting or suitable: a simple URL change allows such a substitution providing the same protocol is followed by the substituting module.

WebKB has three kinds of components: knowledge/text processors, knowledge bases and interfaces. All the components are WWW-accessible.

Knowledge/Text Processors

WebKB allows its users to search, build, manage or generate knowledge or DEs via a *language of commands*. There are three kinds of commands: (i) shell-like text processing commands — such as cat, grep, awk and diff — but working on WWW-accessible files; (ii) knowledge processing commands (at present, only CG processing commands); and (iii) commands of a simple shell-like scripting language — such as if, for, set, pipe — for combining other commands.

In order to be processed, these commands must be sent to a WebKB CGI server (e.g. from a Web browser via the HTML forms of the WebKB interface or from another software by using the CGI protocol (URL: `http://hoohoo.ncsa.uiuc.edu/cgi/`)). Commands (or scripts of commands) may be stored in files and may be mixed with other DEs. To separate a sequence of commands from the rest of the document, the sequence must be enclosed either by the special HTML tag "`<KR>`" and "`</KR>`", or by the strings "$(" and ")$".

Interfaces

Whenever possible, we have used HTML and Javascript to implement the interfaces of our tools. Thus, users may customize sources. As opposed to CGI servers (servers using the Common Gateway Interface), Javascript programs are directly executed on client machines.

For the CG graphic editor (WebKB-GE [15]), where substantial direct manipulation of graphics are required, Java has been used.

The current interface tools of WebKB are the following: a textual CG editor and a graphic CG editor; an ontology editor for the edition of type definitions or relations between types and/or individuals, and a hierarchy browser for such relations.

Knowledge bases and ontologies

Existing top-level ontologies, domain ontologies or even more complete knowledge bases in a relevant domain can be reused to guide knowledge modeling: they show what kind of information might be modelled (and thus what kind of information to collect in documents), and how to model it. They also spare considerable design work.

WebKB provides a top-level ontology of about 200 concept types and 200 relation types (some of the uppermost concept types are shown in Fig. 10). This ontology was created by merging other current top-level ontologies used in the knowledge acquisition and knowledge representation literature and cooperation-oriented hypertext tools. For example, the relation type ontology collects thematic, mathematical, spatial, temporal, rhetoric and argumentative relations types.

Constraints associated to types in our top-level ontology help to increase the consistency of knowledge representations. For example, exclusion links between types ensures that exclusive types such as Spatial-entity, Information-entity and Process never have common subtypes. We have noticed from experience that such a safeguard is extremely useful especially because the subsumption relation is ill-employed, for example when defining a part or a role of a kind of entity instead of a more specialized kind, or because a user may misinterpret or forget the category to which a type pertains, e.g. a type named “union” by a user to refer to an organization could be interpreted by another user as refering to a process, a physical entity resulting from a process, a state, and even a location; our top-level ontology prevents such misinterpretations. The signatures we have associated with our relation types also prevents similar misinterpretations during an ontology extension but also during the construction or use of CGs.

At present, user knowledge, including ontologies, can only be stored in documents. When a warehouse is used, it will be initialises with our top-level ontology, specialized by the 90,000 categories of the natural language ontology WordNet[14]. Thus knowledge modeling is eased since, in most domains and applications, users will find a lot of relevant concept type to specialize and a lot of relations signed on this vocabularies. Knowledge sharing and reuse will also be eased since knowledge from different users will derive from a common vocabulary. Knowledge retrieval is enhanced because it will be possible to use a lot of natural language concept types (and use different synonym names to refer to them) for accessing knowledge. We will also allow this for retrieving knowledge stored in documents but with far less precision since the types they use are not related to the types in the warehouse (unless derived

```
Something
  Something > Entity Situation
  Entity > Description Physical_entity;
    Physical_entity > Cat Mat Table Person;
    Description > Hypothesis;
    Situation > Process State;
    Process > Believe;
```

Figure 10: Defining a simple ontology.

from our top-level ontology, in which case some good guess can be made by exploiting the constraints on the signature relations). We have already exploited WordNet in such directions in our previous tool, CGKAT[11].

V. EXAMPLES OF WEBKB

To experiment with WebKB the reader should visit
`http://www.int.gu.edu.au/kvo/software`
with a Javascript enabled web browser. Select “WebKB”. The *Tool to index DEs by knowledge representations* provides a form entry identifying the URL for the document, the specific document element and its representation as a conceptual graph.

Fig. 2 shows an indexation of a sentence “John is repairing his own car”. The WebKB *Tool to index DEs by knowledge representations* provides a form entry for helping to build such indexations. It asks for the URL of the indexed document, the specific document element and its representation as a CG. Once the form is complete, the user may ask for the generation of the indexation in the WebKB language, which may then be saved on the user local disk or copy/pasted in another document.

The textual or graphic CG editors ease the creation of CGs. (Figs. 6–9 were created using WebKB-GE [15], the graphical editor). The types used in the CGs may be retrieved and selected via the hierarchy browser. Subsumption relations and exclusion relation between types may also be specified using the WebKB language (as in Fig. 10) or graphically via WebKB-GE.

Having assembled all the elements required by WebKB to perform knowledge retrieval, we now call *Information Retrieval/Handling tools* from the main WebKB menu. A form-based interface to the knowledge engine is then presented. An example under *Examples of queries and scripts* is shown in Fig 11. This example can be cut and paste into the query field and submitted to the inference engine. Firstly, it loads the knowledge annotated `exInterviewIndexation.html`. The inference engine is asked to use this knowledge base in order to answer the query *“Tell me about all the specializations of tasks that are subtasks of others?”*, this is written
`spec [Task]->(Subtask)->[Task]`. A new Netscape window will appear with the query result. The result of the query permits the user to navigate both the knowledge base, as knowledge annotations, as well as the original documents from which the annotations were derived.

Similar queries and outputs will result from submitting
spec [Vehicle] and
`spec Something_related_to_road_accident`. Note also that options include both knowledge and source, or source, or knowledge only. The example output is itself a document con-

```
          /WebKBtools/exInterviewIndexation.html;
  spec [Task]->(Subtask)->[Task];
  spec [Vehicle];
  spec Something_related_to_road_accident;
```

Figure 11: Loading an indexation file to test specialization.

taining both knowledge and the original source. It can be saved and thus demonstrates how WebKB enables the construction of documents using knowledge inference to assemble document elements.

WebKB has a broad range of applications: precision-oriented IR, Knowledge Acquisition (KA) and Computer Supported Cooperative Work (CSCW). The two most important facilities for KA and CSCW that WebKB is aimed is to provide knowledge-based comparison and synthesis of information provided by different authors, and user control of generated document content, form or sequence via document descriptions using HTML, Javascript and conceptual queries.

VI. CONCLUSION

WebKB is an indexation tool representing a substantial investment in programming effort. Its motivations are as a public domain research tool to experiment with knowledge retrieval on the WWW. It is intended to combine various technologies for helping Knowledge Acquisition, Information Retrieval and CSCW, notably WWW-related technologies, databases and the knowledge representation languages and processors.

Many of the ideas contained within WebKB maybe a precursor to the functionality of the next generation of commercial-grade web-based knowledge mark up and information retrieval tools. Our motivation is to use this toolkit to improve productivity in knowledge acquisition and knowledge indexing for several specific projects of interest to the Australian Defence Science Technology Organisation (DSTO). We are also motivated by a demonstration that Conceptual Graphs as a knowledge representation can be used for knowledge interchange and annotation. This paper has shown how DEs can be indexed by knowledge annotations as CGs and subsequently how inference with CGs can be used to generate new documents via navigating the knowledge base of annotations.

References

[1] B. Carbonneill and O. Haemmerlé, ROCK : Un système de Question/Réponse fondé sur le formalisme des Graphes Conceptuels. *In Actes du 9eme congrès Reconnaissance des Formes et Intelligence Artificielle*, pp. 159-169, Paris, January 1994.

[2] G. Ellis, Managing Complex Objects. PhD Thesis, The University of Queensland, Department of Computer Sciences, Australia, 1995.

[3] M.R. Genesereth, Knowledge Interchange Format. *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, Cambridge, MA, pp. 599-600. Morgan Kaufmann, 1991.

[4] M.R. Genesereth and R.E. Fikes, Knowledge Interchange Format, Version 3.0 Reference Manual. *Technical Report Logic-92-1*, Computer Science, Stanford University, 1992.

[5] T.R. Gruber, The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases: in *Principles of Knowl- ternational Conference*, Cambridge, MA, pp. 601-602, Morgan Kaufmann, 1991.

[6] O. Haemmerlé, *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels.* PhD Thesis, Montpellier II University, France

[7] A. Kheirbek and Y. Chiaramella, Integrating Hypermedia and Information Retrieval with Conceptual Graphs: *In Hypertext-Information Retrieval-Multimedia (HIM'95)*, Konstanz, Germany, April, pp. 47-60, 1995.

[8] Y. Chiaramella and A. Kheirbek, An Integrated model for Hypermedia and Information Retrieval, in *Information Retrieval and Hypertext*, M. Agosti and A. Smeaton (Ed), Kluwer Academic, pp. 139-178, 1996.

[9] D. LaLiberté, Collaboration with HyperNews: In *Proceedings of Workshop on WWW and Collaboration*, Cambridge, MA, September 11-12, 1995.

[10] C. Luigi Di Pace, P. Leo and A. Maffione, NetRepository: A Networked Information Repository which Supplies Ontologies for Retrieving Information, in *Conceptual Structures: Fulfilling Peirce's Dream*, pp. 145-159, Springer Verlag, LNAI 1114, 1997, Berlin.

[11] P. Martin, Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition. *Proceedings of Peirce'95, 4th International Workshop on Peirce*, University of California, Santa Cruz, August 18, 1995.

[12] P. 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, October 14, 1996.

[13] P. Martin, The WebKB set of tools: a common scheme for shared WWW Annotations, shared knowledge bases and information retrieval, *Proceedings of the 5th International Conference on Conceptual Structures (ICCS'97)*, Springer Verlag, LNAI 1257, Seattle, August 4-8, pp. 585-588, 1997.

[14] G.A. Beckwith, C. Fellbaum, D. Gross and K. J. Miller., Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, Vol. 3, No. 4, pp. 235-244, 1990.

[15] S. Pollitt, A. Burrow and P. Eklund, .In: M.L. Mugnier and J.F. Sowa (eds,): *Proceedings of the 5th International Conference on Conceptual Structures*: Springer Verlag, August 1998.

[16] A. Puder, S. Markwitz and F. Gudermann, Service Trading Using Conceptual Structures. In: Conceptual Structures: Applications, Implementations and Theory: pp. 59-73, Springer Verlag, LNAI 994, 1995.

[17] G. Rodríguez and L. Navarro, AlephWeb: a CSCW Large Scale Trader. *www.pangea.org/alephweb.aleph/paper.html*

[18] M. Röscheisen, C. Mogensen and T. Winograd, Beyond Browsing: Shared Comments, SOAPs, Trails, and On-line Communities, *In Proceedings of the Third International World-Wide Web Conference*, Darmstadt, Germany, April 1995.

[19] J.F. Sowa, Conceptual Graphs Summary, In: Nagle, T.E., Nagle, J.A., Gerholz, L.L., and P.W. Eklund (eds,): *Conceptual Structures: Current Research and Practice*, Ellis Horwood, 1992, pp. 3-51.