

Logic-based Networks: Concept Graphs and Conceptual Structures

Peter W. Eklund

Knowledge, Visualization and Ordering Laboratory
School of Information Technology, Griffith University
Parklands Drive, Southport, Queensland 9726

AUSTRALIA

p.eklund@gu.edu.au

Abstract. Logic-based networks are semantic networks that support reasoning capabilities. In this paper, knowledge processing within logic-based networks is viewed as three stages. The first stage involves the formation of concepts and relations: the basic primitives with which we wish to formulate knowledge. The second stage involves the formation of well-formed formulas that express knowledge about the primitive concepts and relations once isolated. The final stage involves efficiently processing the wffs to the desired end. Our research involves each of these steps as they relate to Sowa's *conceptual structures* and Wille's *concept lattices*. Formal Concept Analysis gives us a capability to perform concept formation via symbolic machine learning. Concept(ual) Graphs provide a means to describe relational properties between primitive concept and relation types. Finally, techniques from other areas of computer science are required to compute logic-based networks efficiently. This paper illustrates the three stages of knowledge processing in practical terms using examples from our research.

1 Introduction

The research report in this paper investigates the viability of conceptual graphs (CGs [24, 25]) as a knowledge representation in general [20, 10, 12, 18, 19], its effectiveness as a graphical aid to cognition [10, 11, 2, 3], and how a formal theory of order can be used to compute conceptual structures in an efficient way [9, 7].

This paper is structured in five sections. I demonstrate the use of formal concept analysis (FCA) by way of a project for developing a medical retrieval system in Section 2. There (and again in Section 3) I show how FCA can be used to efficiently visualize and filter a document collection [6]. The approach re-uses a medical thesaurus as a collection of attributes extracted from a document corpus. Each document is in turn treated as an object. In small attribute collections this approach has merit[4] but it becomes visually intractable when large numbers of attributes are used. Section 2 shows how we can relieve visual complexity while preserving the efficiency of the lattice-based approach to document filtering. This work, extended in Section 3 to filter email [7], is more related to FCA, but does have its analogue in concept(ual) graphs and a more complete version of this relationship is reported in these proceedings¹.

¹ see "CEM – A Conceptual email Manager", R. Cole and G. Stumme in this volume.

In Section 4, I illustrate the practical outcomes of the work of Prediger and Wille [22] on concept graphs. This theoretical framework was coded as software by Bernd Groh and reported in [15]. Groh’s software demonstrates how Wille’s Power Context Family (PCF) can be interpreted as concept(ual) graphs and provides us with an application that integrates FCA, relational databases and a query interface using conceptual structures². Finally, in Section 5, a Web-accessible knowledge representation toolkit is profiled [18, 19]. Our claim is that WebKB was the world’s first precision Web-based information retrieval engine using conceptual graphs and many of the lessons learned in its development are relevant to emerging Web standards such as XML/RDF.

```

{{ Problem List }
 { 0. Small cell ca left lung, 8 cycles chemotherapy 1990, plus
  radiotherapy 1991. 1. Pulmonary embolus. 2. Glaucoma.
  3. Peptic ulcer. 4. Cholecystectomy. 5. Appendicectomy.
  6. Oophorectomy. 7. Right sided pneumonia and neutropaenia. }}
{{ Discharge Treatment }
 { Coloxyl with senna 2 tabs bd, Ventolin 90 sec 4 hrly prn,
  Ranitidine 300mg bd, Mylanta 20 mls tds prn,
  Nifedipine 10mg tds, Panadeine forte 1-2 tabs 4 hrly prn,
  Dipiverine Hydrochloride .1% 2 dps bd both eyes. }}
{{ Information to Patient }
 { Patient aware of diagnosis and limited prognosis.
  Knows to present to LMO or RAH with any problems. }}
{{ Summary of Admission }
 { 68 year old woman, well known to S2. Discharged one week ago.
  Day after discharge, developed increasing SOB with
  yellow/white sputum production. Felt unwell, but denies
  rigors or chills. Using Ventolin regularly with no
  improvement. Transferred by ambulance to RAH. }}
{{ Examination }
 { ; mildly tachypnoeac, RR 30, not cyanosed, looks unwell,
  febrile 38.9, dehydrated, HR 120/regular, BP 140/90, JVP RR,
  HS dual + nil, no ankle swelling, peripheral pulses all
  present, TML, PH dull left base, BS vesicular, reduced
  at left base, crackles right anterior chest. Abdominal
  and neurological examination unremarkable. }}
{{ Investigation }
 { }}
{{ Progress }
 { a steady improvement made. Freely mobile around the ward
  without oxygen on discharge. }}
{{ Follow Up }
 { Chest Clinic/Dr. Holmes LMO to perform MSA20 prior to GPD
  appt. }}
{{ Copies }
 { lmo,file,ur. }}

```

Fig. 1. A typical medical discharge document with the sub-headings — Problems List, Treatment, Patient Information, Admission, Examination, Investigation, Progress and Follow up.

2 An Application of Formal Concept Analysis

The first stage of knowledge processing within logic-based networks involves the formation of concepts, i.e. the basic primitives such as objects, attributes, relations, and concepts to formulate knowledge. For the formalization of those primitives, we adopted the approach of *Formal Concept Analysis* (FCA). FCA has been developed during the last twenty years and has already successfully applied to knowledge processing [27]. The Mathematics of FCA has been described in Ganter and Wille [14].

² see P. Eklund, B. Groh, G. Stumme and R. Wille, “A Contextual-Logic Extension of TOSCANA”— elsewhere in this volume.

	small cell carcinoma	nasal spray	hypertension	terazepam	chronic cough	obstruction	pulmonary	racitidine	peptic ulcer	chemotherapy	infection	glaucom
d1	X	X			X	X	X				X	X	X	X	X		X
d2		X															
d3	X						X				X	X					
d4		X			X								X		X		X
....																	
....																	
d20	X						X				X	X		X			
....																	
....																	
d100	X				X	X	X				X	X		X			

Fig. 2. An abbreviated view of the document context, the objects are document numbers written down the rows and attributes are stemmed MeSH terms written across the columns. In this example there are 100 documents in the context. In the actual context there are slightly less than 4,000 documents and more than 150,000 MeSH terms.

Our first experience using FCA methods was to develop of a medical retrieval system in cooperation with the Royal Adelaide Hospital. Using 4,000 patient discharge documents, one of which is shown below in Fig 1, we found a number of electronic medical thesaurus and built a data context with documents as objects and medical terms as attributes: shown in Fig. 2. The project can therefore be considered as a logic-based network where the concepts are the medical terms and the relations and occurrences of those terms.

Term identification and extraction from texts is a difficult problem. One of the first tasks is to semi-automate term identification. Fig. 4 shows an interface written for the clinician to located and extract MeSH [1] terms from the documents. The top left frame of Fig. 4 shows a document, the interface to the top-right allows a text string to be entered, cross-referenced with aliased MeSH terms, and located in the text. The clinician is required to confirm likely matches, abbreviations and close spellings: once these identified they are later recalled. This idea can be further extended to hyperlinking MeSH terms to navigate the document collection[16]. If we are interested in documents that contain the terms “carcinoma” and “Drug Therapy <1>” for example, the first document in this set of 214 is shown in Fig. 5.

One of the principles of our work is the human centered nature of reasoning and decision making. We achieve a dramatic reduction in the complexity of both algorithmic and visualization components by placing the human operator at the center of the information filtering process. The human selects a theme and this determines the output (these themes are called *conceptual scales* in the FCA literature). This idea is shown in Fig 6. With respect to logic-based networks

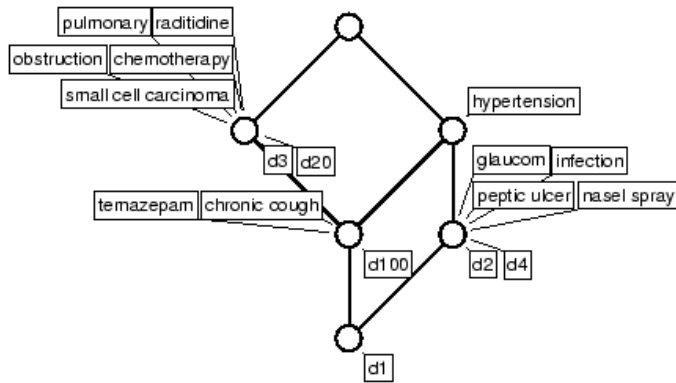


Fig. 3. A conceptual scale can be thought of as a *theme* that reduces the display and computational complexity. This figure shows a conceptual scale evident in the formal context shown in Fig. 2.

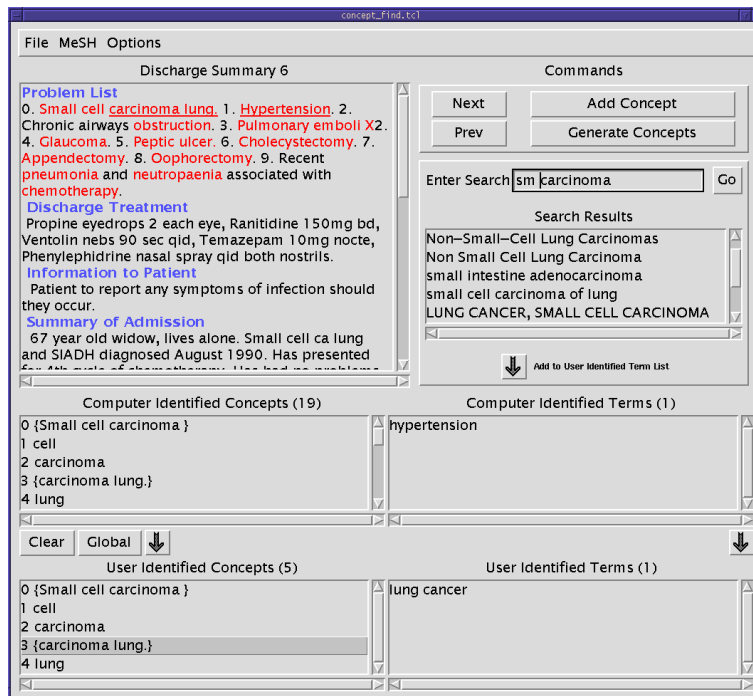


Fig. 4. The MeSH concept extraction program — shows the text (top-left) together with the MeSH terms that are contained within it (middle-left). The program was written by Richard Cole in TCL/TK.

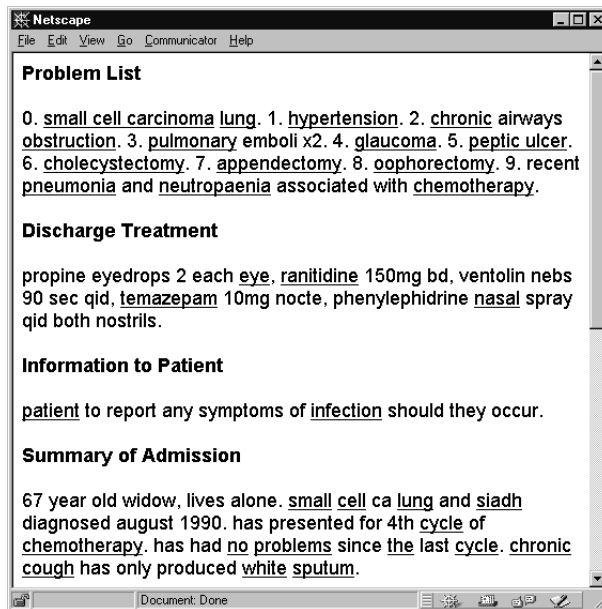


Fig. 5. First of 214 generated HTML documents hyper-linked on the MeSH terms “carcinoma” and “Drug Therapy <1>”.

the concepts are the MeSH terms (and their synonyms) and the relations are the presence (or absence) of Mesh terms in the text. Fig. 4 is the interface that permits these relations to be asserted. Both the computer and the user are engaged in the process of concept identification.

Placing the human at the center of the information flow is obvious but often overlooked in intelligent systems. By asking the user to refine the scope of the exploration by defining a conceptual scale (theme) the complexity of the problem and its representation are profoundly decreased. In some ways the results of our work with medical texts are inconclusive. As an information retrieval metaphor, the concept lattices and conceptual scales have not been bench-marked against more established IR techniques but the intuitions are promising. Complexity can therefore be reduced by considering the following;

1. reducing the attribute sets M to \underline{M} by selecting only those attributes representing a “focus” of interest. This is described as a user-defined *theme* or *conceptual scale*.
2. by reducing the size of the object set G to \underline{G} by collapsing documents into equivalent classes, i.e. if two or more documents contain the same set of attributes they can be considered as a single equivalence class of documents.
3. the analysis of the distribution of attributes in the document space can lead to efficient encoding techniques that reduce the space complexity of the representation.

Our emphasis is on visual outcomes used for text data mining. Showing that the visual complexity of the lattice representation can be used to explore

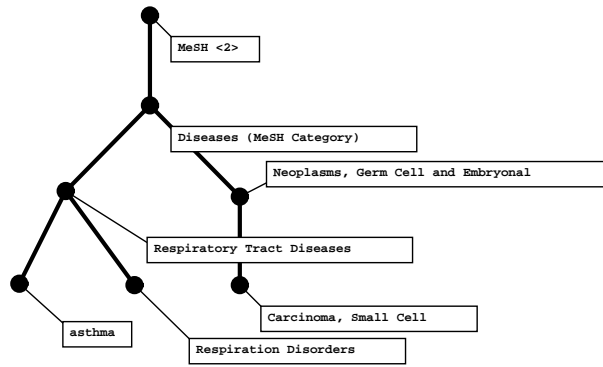


Fig. 6. Folding Poset Viewer with MeSH Concepts, displaying the theme created by the user, in this case exploring a possible relationship between asthma and carcinoma. The scale partitions the document collection on the terms nominated in the theme.

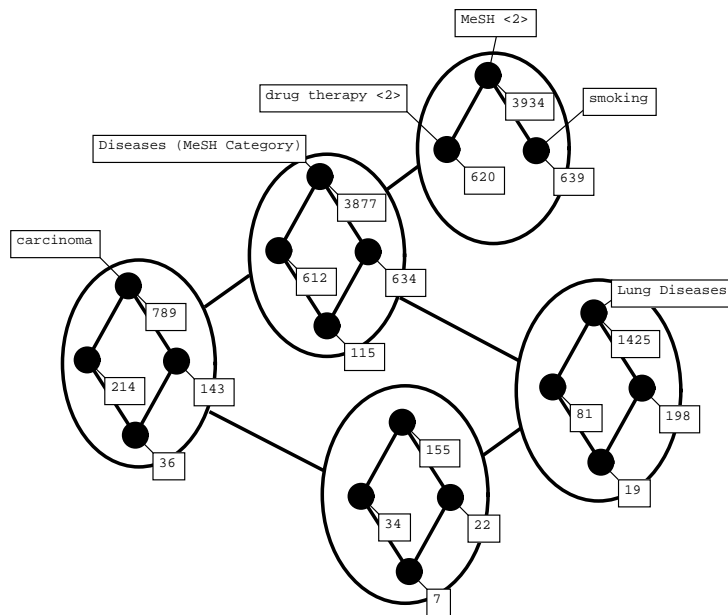


Fig. 7. A nested line diagram exploring the relationship between Carcinoma, Chemotherapy and Smoking. The numbers attached to vertices indicate the number of documents that cluster to that vertex.

a document collection in an intuitive human-centered interface is an important sub-goal. The graphical interface allows key terms to be identified from a medical dictionary of terms. It builds a theme (conceptual scale), the concept lattice which is a \vee -homomorphic image of the whole data context. Fig. 7 shows a nested line diagram of a simple scale exploring the relationship between smoking and carcinoma.

Feedback on the creation of the conceptual scale is provided in a number of ways. All synonyms of the key term to be scaled are shown and the number of documents containing a key term are displayed — this gives feedback on the uniqueness and relevance of the search term. For any given term, its children can be listed as well as its parents — this gives the user an easy way of constraining search to include more documents through the use of a more general term, or alternatively, specialize a term to focus the search on fewer documents. Once the conceptual scale has been defined it may be saved together with an appropriate labelled line diagram of its concept lattice. W.r.t. logic-based networks, the conceptual scale is like a well-formed formula. The metaphor here is that the scale can only be a structural combination of concepts organised hierarchically. When two terms are added to the scale, their least upper bound is also added.

The presentation of the lattice is important and some specific lattice drawing algorithms can be imported from the graph layout literature [8, 26] although this issue is far from conclusively researched [5]. Of further aid is the progressive (un)folding of the finite lattice, this is user-controlled, but provides a clear mechanism for reducing the display complexity.

3 Email Concept Analysis (CEM)

A naturally extension of the application of FCA to document processing is to look at emails as documents. The CEM program [6] follows from the medical document system described above and embodies much of the elements our research philosophy: it is useful, it relies on humans to boot-strap its knowledge by creating a conceptual ontology of terms and definitions and how these terms manifest. It is flexible, it allows the operator to design his theme or view. It is efficient, it uses fast algorithms that compute in polynomial time [7]. Unlike the medical document project the terms are user-defined types. These types have a “classifier” attached that tells the information retrieval system how to associate a document to a type.

The CEM program is intended as a framework to create a visual and flexible view over email. Its purpose is to pose questions about the ways in which keywords combine in the email in order to extract (and find) interesting information. The CEM program is based on a *formal context* (G, M, I) where: the object set G consists of emails, the attribute set M consists of terms for emails, and I is the relation between emails and their assigned terms.

$a \rightarrow b$ is used to denote the implication determined by a formal context that: every object of the context that has attribute a will also have attribute b . In the case of email, we allow the user to state a set of implications that are enforced on the context. The implications are expressed by an order relation \leq giving rise to an ordered attribute set (M, \leq) (the order relation is reflexive, transitive,

and anti-symmetric). A formal context, $\langle G, M, I \rangle$, is said to respect the set of implications given by (M, \leq) if $(g, m) \in I$ and $m \leq n$ always imply $(g, n) \in I$. Consequently, if an email g has attribute m and the user has indicated that $m \leq n$ then the email will also have attribute n .

Each of the attributes may be associated with a classifier for automatically associating emails. Two examples of these are shown below. An email is said to be of the type “From Eklund” if it contains in the email “from” field the term “p.eklund@gu.edu.au”. The type “KVO Meeting” is an email from “p.eklund@gu.edu.au” with the email subject “KVO Meeting”.

"From Eklund"	From: p.eklund@gu.edu.au
"KVO Meeting"	From: p.eklund@gu.edu.au
	Subject: KVO Meeting

The user is asked to create an implication ordering $<$ between admissible attributes.

"From Eklund" $<$ "From KVO Group"
 "From KVO Group" $<$ "KVO Group"

In the logic-based network framework, the concepts are the types, the relations between the concepts and documents is determined by the user-defined “classifier”. The type hierarchy gives us a way of formulating a coherent and connected theme/scale over the email. Inference patterns are “suggested” by the cardinality of emails that cluster to a particular vertex in the lattice. The reflexive transitive closure of the statements made by the user gives (M, \leq) .

A conceptual scale (or theme) is determined by a set of types selected by the user. The user adds types to the conceptual view by text searching over the names of the types. When a type is selected it is added to the diagram showing the hierarchical ordering of the types. The diagram is completed by all types (ordered in advance) greater than a selected type. For instance, selecting the attribute “From Peter Deer” in Fig. 8 (top-left) forces the addition of “DSTO”.

The concept lattice of the conceptual scale is shown below the chosen hierarchical attribute order in Fig. 8; the filled circles represent concepts completing the hierarchy of attribute concepts to obtain the whole lattice. The extent of the infimum of a selection of attribute concepts (example: “From Peter Deer”, “Mention Richard”) consists just of all objects having the selected attributes; the number of those objects is attached to the circle representing the infimum (24 in the lower-left of Fig. 8). In the diagram of Fig. 8 (bottom-left), the extent numbers show that there are 2,640 emails that are related to some research group. These 2,640 is split into 425 related to “DSTO” and 2,278 to “KVO”, 166 emails are related to both.

The concept lattice may be large and difficult to comprehend. A successful method to draw larger concept lattices is representation by nested line diagrams. This is grounded on the construction of direct products of lattices. To draw a concept lattice using a lattice product we divide the attribute set M_s (user selected) into two sets M_1 and M_2 . The two attribute sets can then be used to construct two concept lattices $\underline{\mathfrak{B}}(G, M_1, I \cap G \times M_1)$ and $\underline{\mathfrak{B}}(G, M_2, I \cap G \times M_2)$. Then the following mapping defines a \vee -preserving embedding of

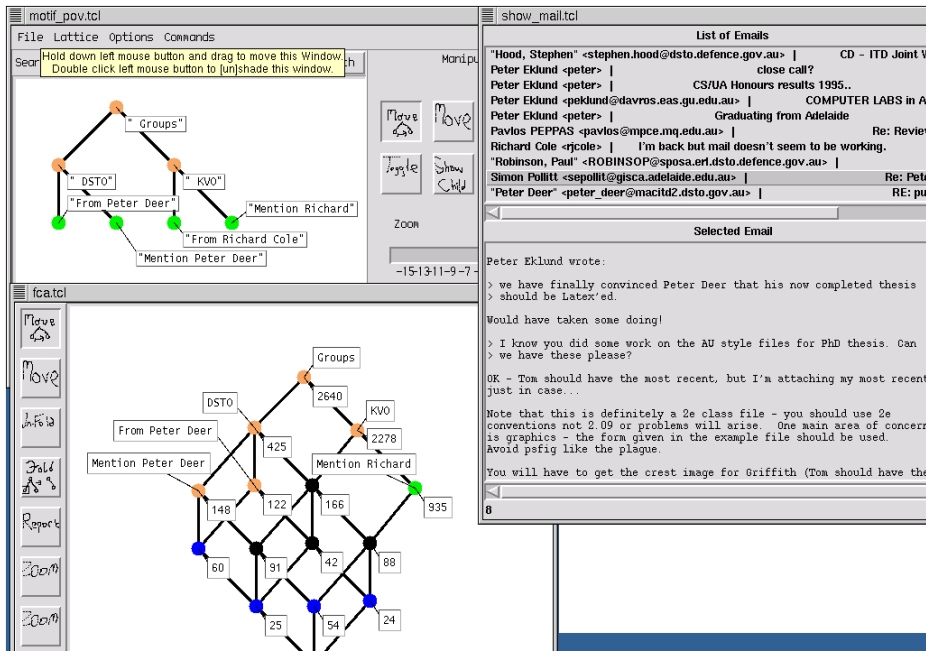


Fig. 8. The CEM Program Interface. Top-left is the theme editor, bottom-left the concept lattice resulting from the theme. On the top-right email headers and lower-right the email contents.

$\mathfrak{B}(G, M_s, I)$ into the direct product of the two constructed concept lattices: $(A, B) \mapsto ((B \cap M_1)', B \cap M_1), (B \cap M_2)', B \cap M_2)$. In Fig. 9, a nested line diagram of $\mathfrak{B}(G, M_s, I)$ is shown. The elements of the first concept lattice with $M_1 := \{\text{"Groups"}, \text{"DSTC"}, \text{"From Melfyn"}, \text{"Mention Melfyn"}\}$ are represented by large circle each of which contains a copy of the diagram for the second concept lattice with $M_2 := \{\text{"top"}, \text{"From KVO"}, \text{"From Bernd Groh"}, \text{"From Peter Eklund"}\}$, "WebKB"}.

This nesting of the line diagrams of the two lattices is a graphical representation of the direct product of the lattices. The elements of the lattice product that are not mapped by concepts of $\mathfrak{B}(G, M_s, I)$ are shown by grey circles. Grey circles indicate implications that come from the data rather than from the hierarchy, i.e. implicitly present in the data but not in the intended theory. CEM uses an user-oriented theory as a framework for deductive inferences. This framework conditions the presentation of the material but it also uses a data driven approach for imputation and induction.

In Fig. 9, where the outer scale is M_1 and the inner scale M_2 , there are 113 emails that mention Melfyn (lower-left ellipse) and 569 that are from "Peter Eklund" (middle vertex in upper-left ellipse). Of the 113 emails that mention Melfyn there are 48 that are also "From Peter Eklund" (middle vertex in lower-left ellipse). Of the 113 emails that mention Melfyn 48 (almost half) come from Peter Eklund. Melfyn is mentioned mostly by Peter Eklund and Melfyn (likely

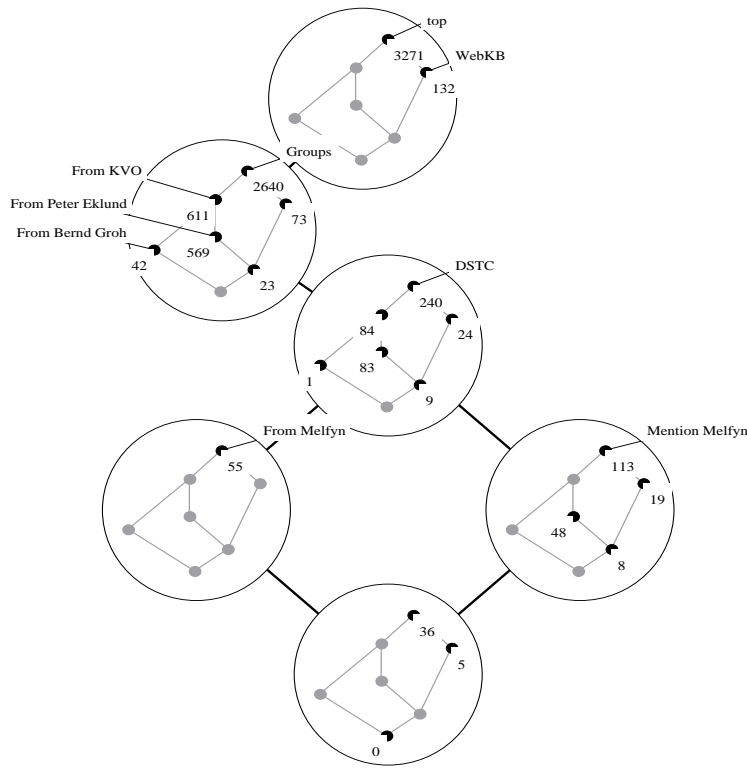


Fig. 9. Nested Line Diagrams reduce visual complexity.

as a signature). Over half the email sent by Peter Eklund concerning the DTSC mention “Melfyn”. These are some of the simple inferences that can be read from the nexted line diagram in this simple example.

4 Concept(ual) Graphs - CGPCF

Conceptual Graphs (CG) can be used to describe more completely facts about the world but they have sometimes lacked a precise semantics and a scalable implementation. Wille developed the theory of a *power context families* (PCFs) that presents *concept graphs* as mathematical algebraic entities called PCFs. Bernd Groh developed an algorithm that can derive concept graphs from PCFs [15]. Bernd implemented this theory and released it as a CG implementation with a back-end relational database management system shown in Fig. 10.

CGPCF is another example of the convergence of theoretical and empirical theory building, it is neither a general purpose programming language nor an application program, it is a powerful, scalable and general technique for storing, searching and retrieving complex data formulated as a knowledge-base.

From the point of view of logic-based networks, we have come up with a data structure that contains explicit representations of concepts and relations connecting concepts. Relations are therefore no longer user-determined but rather user-defined. The problems that this imposes are many. If relations between concepts

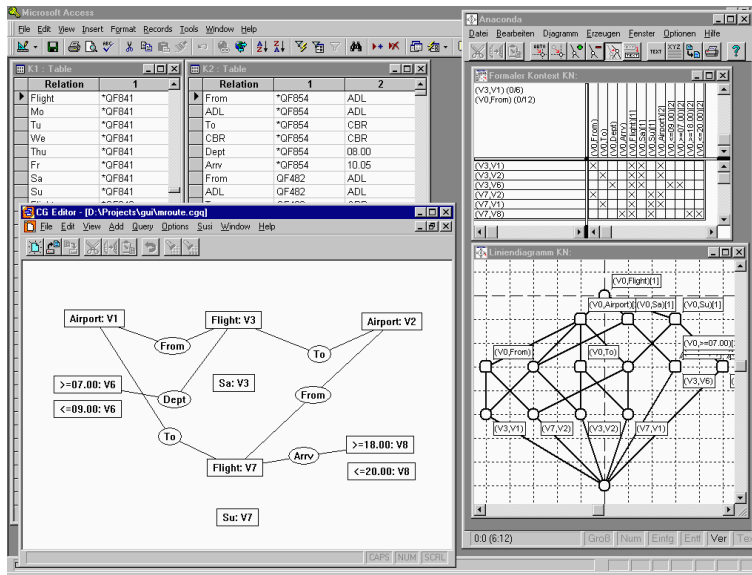


Fig. 10. Bernd Groh's CGPCF TOSCANA integration. The lower left shows a CG, top-left the relational database formulation of a schedule (as knowledge-base), top-right cross-table as a Power Context Family (PCF), and bottom-right the formal concept lattice of the cross-table.

are explicitly contained with the text we need mechanisms that facilitate their extraction. The technical question is how to generate well-formed formulas from the PCF and how to use those formulas for inference. This is a current area of investigation.

5 WebKB

In terms of logic-based networks we now come full circle. WEBKB [18, 19] is an important system because it is the first of its type that allows knowledge to be described from the Web. In this sense it is years ahead of the emerging Web standards in XML that will permit knowledge embedded into Web documents.

WEBKB is a system for annotating document elements in Web-pages with conceptual graphs that are representative of semantic meaning. This is an activity that in WEBKB is done by human hand. However much of what we have learned in this paper about logic-based networks suggests that it may be possible to generate conceptual graphs automatically from Web documents using some combination of type and relation hierarchies & concept and term identification techniques: storing this knowledge as a power context families from which a CG can automatically be derived. Once derived the CGs can be used to infer more knowledge. We call this idea of bringing the semantic content of two or more documents together *knowledge fusion*. The result of this idea is a program called HIBKB (currently under development which I will demo at ICCS2000). In the meanwhile it is worthwhile reviewing WEBKB, since it demonstrates a

high-level inferencing system for CGs and how such a system can hyperlink a knowledge-base and document collection.

Retrieving semantic content on the Web is an open research problem. Despite this, some of the infrastructure that supports this task is becoming available. Manually structuring Web documents using syntactic mark-up languages such as XML³ allows Web-robots to retrieve relatively precise information using string and structure-matching techniques. However, the Web robot approach is not scalable because fine-grained information is retrieved only when documents are richly structured and the querier understands this structure: the exact tag names and their order. Using knowledge representation languages as meta-data is a solution to the problem but it imposes several requirements on the knowledge representation language.

A first requirement is that the notation be intuitive and concise enough to be read and understood by people. Most current knowledge-oriented metadata languages are built on top of XML, e.g. RDF⁴ and OML⁵. Choosing XML ensures that standard XML-based tools can parse and exchange metadata. However, since XML is verbose, the metadata languages built on top of XML are verbose, and this makes their use difficult without the aid of specialized editors. Standard XML tools are of little interest in managing knowledge-oriented meta-languages since specialized editors, analyzers and inference engines are required.

Another requirement is that a document's author should be allowed to render knowledge statements visible to the reader. This is particularly true when a graphic can be made with a visual language⁶, or sentences written using a controlled language⁷ — a subset of natural language that eliminates ambiguity. Then, for example, a knowledge base and its associated documentation can be integrated within the same document and both accessed and managed using classic information retrieval techniques (string-search, structure navigation via table of contents, etc) as well as knowledge-based techniques.

Another requirement of a metadata language for knowledge retrieval is that it should be both precise and general enough to allow users to represent any Web-accessible information at a desired level of precision. This means the metadata language should have an expressive formal model. Any formalism equivalent to first-order logic and permitting the use of nested "contexts" is an appropriate candidate, e.g. the Knowledge Interchange Format (KIF)⁸ or Conceptual Graphs (CGs)⁹ [24]. It is important not to restrict the expressivity of the language but it is reasonable to forgo some language features in return for efficiency. This means that an inference engine may ignore some features of the knowledge representation language in the same way that a HTML browser can exploit

³ <http://www.w3.org/XML/>

⁴ <http://www.w3.org/RDF/>

⁵ <http://wave.eecs.wsu.edu/CKRMI/OML.html>

⁶ <http://www.cpsc.ualgary.ca/~kremer/home.html>

⁷ <http://www-wilots.let.uu.nl/Controlled-languages/>

⁸ <http://logic.stanford.edu/kif/kif.html>

⁹ <http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/CGs.html>

some feature tags but not others. For example, a CG-based search engine may ignore references to mathematical sets while exhibiting adequate precision. The RDF metadata language and Ontobroker are general but oriented towards the representation of entire documents (not arbitrary document elements) and do not yet have conventions to represent logic-based features, e.g. quantifiers and logical operators. This limits their current capacity for knowledge sharing on the Web.

In summary, the first three requirements for *precise, flexible and scalable* knowledge and information retrieval implies firstly an easy to use notation that is intuitive, precise and expressive and secondly, the capacity to insert knowledge anywhere within a Web document. WEBKB¹⁰ [18] responds to these requirements. It interprets knowledge statements stored within Web-accessible documents. Each group of statements is delimited by the tags (“<KR>” and “</KR>”) or the strings “\$(” and “)\$”. The statements are visible unless the document’s author hides them within HTML comment tags. The knowledge representation language is specified with a tag attribute, e.g. “<KR language=“CG”>”. WEBKB interprets the linear notation of CGs as well as some simpler linear notations we have invented: a formalized English, a frame-like notation for CGs, and structures that relate document elements by semantic relations¹¹. These simpler notations are translated into the linear CG notation.

CGs were selected because they have a graphical notation and a linear notation, both concise and relatively intuitive, and secondly because we can reuse two CG inference engines (CoGITO [17] and Peirce [13]). The inference engines exploit subsumption defined between formal terms for calculating specialization between CGs — and therefore between queries and facts in a knowledge base. Hence, statements and queries can be made at varying levels of precision.

Both knowledge and string-based commands prove useful within documents. It is handy to be able to use them within documents themselves and — if desired — have their results inserted in place of commands. In the hypertext literature, this is known as *dynamic linking*, and the generated document as a *dynamic document* or a *virtual document* [21]. Virtual documents have many applications including adapting document content to a user. Web robots, such as Harvest¹², WebSQL¹³, WebLog¹⁴ perform some document generation this way.

WEBKB permits virtual documents by combining lexical, structural and knowledge-based data management by proposing commands for searching and joining CGs, Unix-like file management commands working on Web-accessible documents and a simple Unix shell-like script language to combine commands. These commands may be inserted in documents and sent to WEBKB programs.

¹⁰ <http://meganesia.int.gu.edu.au/~phmartin/WebKB/>

¹¹ See “Conventions and Notations for Knowledge Representation and Retrieval”, P. Martin in these proceedings.

¹² <http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/schwartz.harvest/schwartz.harvest.html>

¹³ <http://www.cs.toronto.edu/~websql/>

¹⁴ <http://www.cs.concordia.ca/~special/bibdb/weblog.html>

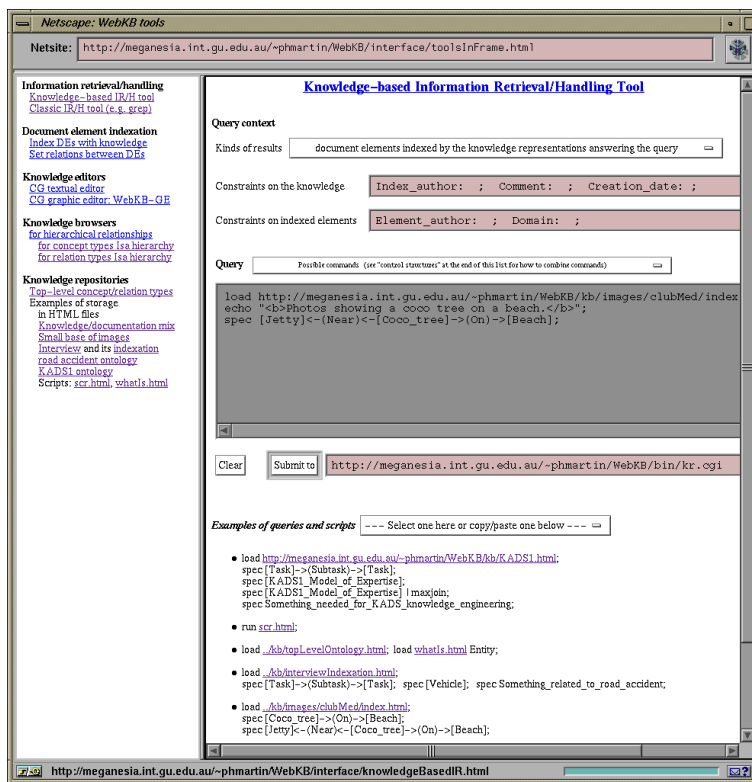


Fig. 11. The WEBKB menu and knowledge-based Information Retrieval/Handling Tool. The example query shows how a document containing CGs (indexing images) is loaded into the WEBKB processor and then how the command `spec` searches for specializations of the CG `[Jetty]<- (Near)<- [Coco-tree]->[On]->[Beach]`.

Fig. 11 shows the WEBKB menu and the “Knowledge-based Information Retrieval/Handling Tool”.

5.1 Representing Knowledge

To compare the alternatives for knowledge representation and retrieval on the Web, Fig. 12 shows how a simple sentence may be represented with CGs, 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 (Fig. 12 top) seems simpler than the others. The semantic network structure of CGs (i.e. typed concepts connected by typed relations) has three advantages. Firstly, it restricts the notation without compromising expressivity — this reduces computational overhead when comparing CGs. Secondly, it encourages users to use explicit relations between concepts, and finally it permits a better visualization of relations between concepts — relations can be rendered as an edge and concepts and a vertex. Even if CGs seem relatively intuitive (they

```

<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/94-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/94-rdf-schema#" xmlns:x="http://www.bar.com/example#"
xmlns:t="http://www.bar.com/topLevelOntology#">
<t:Person bagID="Statement_01"><t:NameMary/></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. 12. Comparing knowledge representation with CGs, KIF and RDF.

are not readable by everyone), simpler notations may be preferred. For instance, Fig. 13 shows other notations accepted by WEBKB.

```

/* Frame CGs */
[Mary; chrc:[an age *a]; cousin:[a person; chrc:*a] ] (Believer: John).

/* Text structured with HTML tags (and same conventions for relations) */
<dl><dt> <dl><dt>Mary<dd>chrc: an age *a
<dd>cousin: <dl><dt>a person<dd>chrc:*a</dl> </dl>
<dd>(believer: John)
</dl>

/* Formalized english */
Mary has for cousin a person
who has for chrc an age chrc of Mary' has for believer John.

```

Fig. 13. Complementary notations for simple knowledge statements.

5.2 Less Expressive, More Intuitive Notations

Users want flexibility. They do not always have time to declare and order some of the terms used when representing knowledge. This is the case when indexation is for private knowledge organisation purposes or when detail is omitted during work in progress. To permit this, and still allow the system to perform some minimal semantic checks, basic declared relation types are used and concept types left undeclared. The rationale for this is that when knowledge statements are made from concepts linked by basic relations, the complexity is concentrated within concept types and only a limited set of relation types are necessary. WEBKB processes 200 basic relation types¹⁵ which collect common thematic,

¹⁵ <http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/topLevelOntology.html>

mathematical, spatial, temporal, rhetorical and argumentative relations types. This collection of basic types is sufficient for most work.

Secondly, WEBKB can use relation signatures to give types to the undeclared terms used as concept types. For instance, in the top-level ontology in 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. WEBKB can infer from this ontology that *Knowledge_design* must be a subtype of *Process* if used with the above relation types.

The ontology of types becomes an important tool for a system like WEBKB. WEBKB controls type signatures and knowledge vocabulary so it needs to be a rich and diverse collection. In WEBKB, we merged the WordNet¹⁶ ontology — 120,000 words linked to 90,000 concept types — into our top-level ontology. Once a WEBKB shared repository is implemented and initialized with these ontologies, it is possible to relate undeclared terms to precise concept types in the repository using the constraints of the relation signatures.

Consider the simple CG $[Cat] \rightarrow (0n) \rightarrow [Table]$. If *Cat* and *Table* have not been declared then the system infers (using WordNet) that *Cat* has 5 meanings (feline, gossip, X-ray, beat and vomit) and *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 a concept of the same type. WEBKB can therefore infer that neither “beat” or “vomit” are the intended meanings for *Cat*, nor “array” or “postpone” for *Table*. To resolve the intended meanings, the system can prompt: “Does *Cat* refer to feline, gossip, X-ray or something else?” and “Does *Table* refer to furniture, tableland, food or something else?”.

Subsumption is an inference that determines whether one knowledge statement can be inferred from another by specialization or generalization. From our previous example, $[Cat] \rightarrow (0n) \rightarrow [object]$ is a generalisation of $[Cat] \rightarrow (0n) \rightarrow [Table]$. Likewise, $[Cat] \rightarrow (0n) \rightarrow [Table] \rightarrow (Attr) - [Legs: \{*\} @4]$, a specialisation of $[Cat] \rightarrow (0n) \rightarrow [Table]$ and $[Cat] \rightarrow (0n) \rightarrow [object]$. This process of comparison is the main mechanism for search and inference in WEBKB. When knowledge statements follow the same conventions they can be readily compared. Using a common and basic relation vocabulary is thus important. A common convention for primitive concepts and complex relations makes comparison more difficult. Take for example the sentence “Mary is 20 years old”. Following our convention a simple relation “(Chrc)” is used, $[Person: "Mary"] \rightarrow (Chrc) \rightarrow [Age: @20]$, the inverse convention would use (Age) as a relation type with the simple concept type $[Integer]$, $[Person: "Mary"] \rightarrow (Age) \rightarrow [Integer: 20]$.

5.3 Indexing Individual Document Elements

We call a Document Element (DE) any textual/HTML data, for example a sentence, a connection, a reference to an image or 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 with knowledge statements, or connect DEs from the same or different documents using relations. Fig. 14 shows an example of each.

¹⁶ <http://www.cogsci.princeton.edu/~wn/>

```

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

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

```

Fig. 14. A language for knowledge indexing or connecting any Web-accessible document element.

The above 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. Fig. 11 presents a general interface for knowledge-based queries and shows how a document containing knowledge is loaded in the WEBKB processor before being queried.

WEBKB also allows the document owner to index an image by a knowledge statement directly stored in the “alt” field of the HTML “img” tag. We use this special case of indexation to present a simple illustration of WEBKB’s features. This example, shown in Fig. 15, is a good synthesis but is 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. Typically, these elements would be split into separate documents. Fig. 16 shows a part of this document that shows the indexation. The result of the query shown in Fig. 15 is displayed in Fig. 17.

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 Harvest, WebSQL or WebLog. Instead, we have implemented Unix-like text processing commands for exploiting Web-accessible documents. The command list includes: `cat`, `grep`, `fgrep`, `diff`, `head`, `tail`, `awk`, `cd`, `pwd`, `wc` and `echo`. A hyperlink path-exploring command “`accessibleDocFrom`” is also provided. 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) including the string “knowledge” in their HTML source code.

```

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

```

WEBKB includes commands for displaying specializations or generalizations of concept and relation types or of an entire CG. At present, queries for CG specializations retrieve only connected CGs: the processor cannot retrieve paths between concepts specified in a query. If a retrieved CG indexes a document element, it can be presented instead of the CG (Fig. 17 gives an example).

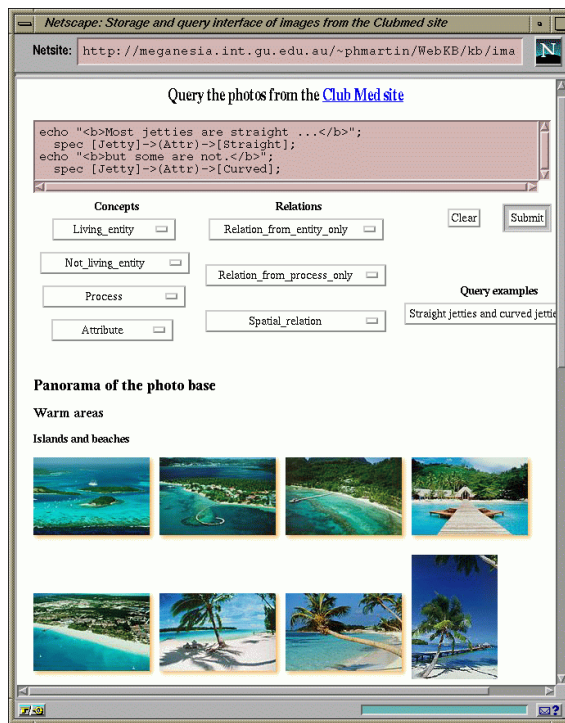


Fig. 15. Images, knowledge indexations and a customized query interface contained within a same document. The example query shows how the command “spec” which looks for specializations of a CG can be used to retrieve images indexed by CGs. The results are shown in Fig. 17).

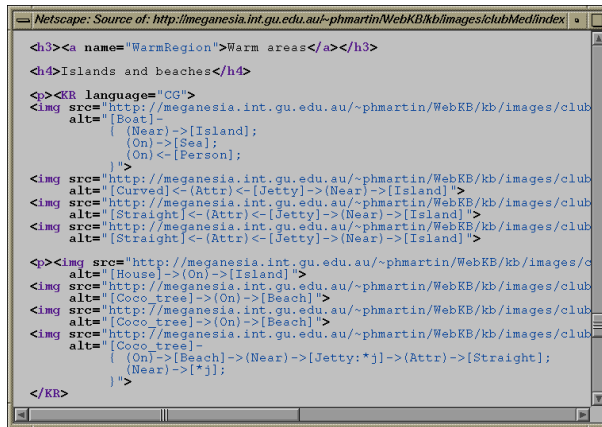


Fig. 16. The HTML source of the indexation of the images shown in Fig. 15.

In both cases, hyper-link 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

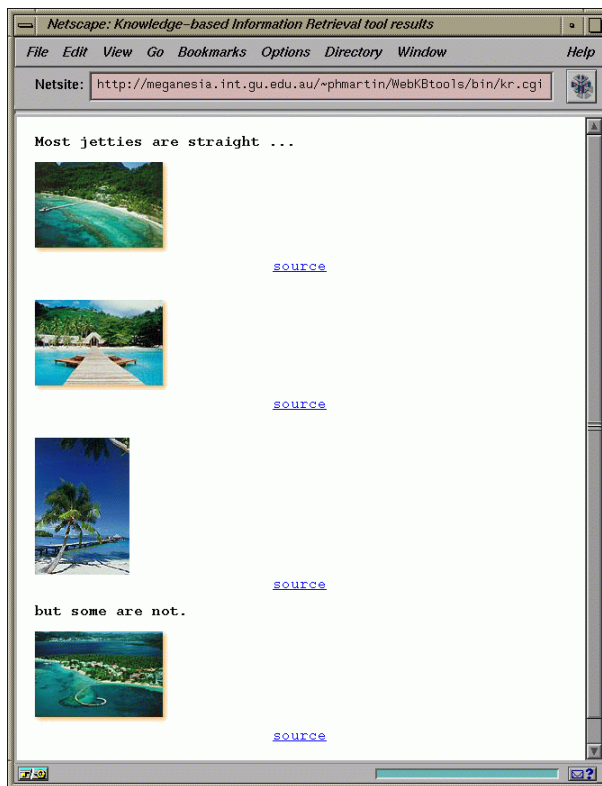


Fig. 17. The document generated in response to the query in Fig. 15.

the indexation in Fig. 14 has been stored, and that *Something* is the most general concept type in the ontology.

```
> 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

Specialization gives the user freedom in the way queries can be formulated: 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 supply only a substring of a type in a query CG, if prefixed by the wildcard character %. WEBKB replaces the substring by declared types including the substring. Replacements violating relation signatures or individual types are discarded. For example, `spec [%thing]` will

trigger the generation and execution of `spec [Something]`, `spec [thingone]` and `spec [thingtwo]`.

Knowledge query commands may be combined with the WEBKB script language to generate complex documents, perform consistency tests on knowledge bases, or solve problems procedurally. The WEBKB site provides many examples of queries and scripts. For example, one script solves a classical resource allocation problem, the Sisyphus-I room allocation problem¹⁷.

One way to generate new knowledge in WEBKB is by joining CGs. Various kinds of joins over CGs may be used but WEBKB only proposes joins which, given a set of CGs, create a new CG specializing each of the source CGs. The result is inserted in the CG base although it may not represent anything true for the user. This does however provide a device for accelerating knowledge representation. For instance, 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]->(0n)->[Mat] [Cat:Tom]->(Near)->[Table]
      [Cat:Tom] - { (0n)->[Mat];
                  (Near)->[Table];
                }
```

Ontology servers, such as the Ontolingua ontology server¹⁸ and Ontosaurus¹⁹, support shared knowledge repositories. However, Ontology servers are not usable for managing large quantities of knowledge and, apart from AI-Trader [23], they do not allow 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 APECKS²⁰, Co4²¹ and Tazebao²².

We are extending WEBKB to handle a knowledge repository. We are considering five issues which address scalability: (i) the implementation of the knowledge-based system reuses FastDB²³, a scalable multi-user persistent object repository, and (ii) algorithms allowing the exploitation of large-scale dynamic taxonomies efficiently²⁴; (iii) visualization techniques (handling term-aliases and view generation) to avoid lexical conflicts and enable users to focus on certain kinds of knowledge; (iv) protocols allowing users to solve semantic conflicts via the insertion of new terms and relations in the common ontology and, in some cases, the knowledge of other users; (v) conventions to improve the automatic comparison of knowledge by different users.

¹⁷ <http://meganesia.int.gu.edu.au/~phmartin/WebKB/kb/sisyphus1.html>

¹⁸ <http://WWW-KSL-SVC.stanford.edu:5915/>

¹⁹ <http://www.isi.edu/isd/ontosaurus.html>

²⁰ <http://www.psychology.nottingham.ac.uk/staff/Jenifer.Tennison/APECKS/>

²¹ <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html>

²² <http://ksi.cpsc.ucalgary.ca:80/KAW/KAW98/domingue/>

²³ <http://www.ispras.ru/knizhnik/fastdb.html>

²⁴ <http://www.cs.sfu.ca/cs/people/GradStudents/fall/personal/index.html>

6 Conclusion

The purpose of this paper has been to profile our research involving concept(ual) graphs. There are some general recommendations that can be drawn: (i) useful intelligent systems result where general frameworks instantiate real application problems; (ii) assumptions about user intentions are minimized when the human operator is placed at the center of the information system. In this way the combinational complexity of an information system can also be minimized; (iii) this in turn facilitates information flows. The hypothesis of our work is that real progress in intelligent system results from these three recommendations.

To this end we have demonstrated a number of intelligent computer systems: a document filtering and retrieval system for medical texts, a similar system for the recovery and analysis of email, a system called CGPCF that fuses formal concept analysis with relational databases and conceptual graphs and our Web-accessible precision-based information retrieval system, WEBKB.

Acknowledgements

Special thanks to members of the KVO laboratory who have prepared the subject matter for this paper: Drs. Philippe Martin, & Francois Modave, Richard Cole, Bernd Groh, Thomas Tilley and Natalyia Roberts. I would also like to thank Prof. Rudolf Wille for his corrections. Any remaining errors are my own.

References

1. *1998 MeSH, Annotated Alphanumeric List*. National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161, 1998.
2. A. L. Burrow and P.W. Eklund. Visual structure representations and conceptual graphs. In *Proceedings of the 3rd International Workshop on PEIRCE*, pages 45–52. ICCS-94, 1994.
3. A.L. Burrow, S.E. Pollitt, and P.W. Eklund. WebKB-GE - A Visual Editor for Canonical Conceptual Graphs. In *Proceedings of the 6th International Conference on Conceptual Structures ICCS '98*, LNAI 1453, pages 111–118. Springer, 1998.
4. C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24:95–122, 1996.
5. R. Cole. Automated layout of concept lattices using force directed placement and genetic algorithms. In J. Edwards, editor, *The 23rd Australasian Computer Science Conference*, volume 22 of *Australian Computer Science Communications*, pages 31–42, Los Alamitos, CA, 2000. IEEE Press.
6. R. Cole and P. Eklund. Analyzing an email collection using formal concept analysis. In *Proceedings of the European Conf. on Knowledge and Data Discovery*, LNAI 1704. Springer, 1999.
7. R. Cole and P. Eklund. Scalability in formal concept analysis. *Computational Intelligence*, 15(1):11–27, 1999.
8. M. K. Coleman and D. Stott-Parker. AGLO — Publications and Implementation. *Software – Practice and Experience*, pages 1415–1438, December 1996.
9. P Eklund. Using path-algebras to encode directed graph knowledge-bases. In *4th Annual Workshop on Conceptual Structures*, Detroit, MN, 1989.
10. P Eklund and J.M. Kellett. Prospects for conceptual graphs in knowledge acquisition interfaces. In *Proceedings of the 3rd European Workshop on Knowledge Acquisition*, 1989.

11. P. Eklund, J. Leane, and C. Nowak. GRIT: A GUI for Conceptual Structures. In *Proceedings of the 2nd International Workshop on PEIRCE*. ICCS-93, 1993.
12. P.W. Eklund and P.H. Martin. WebKB indexation and document retrieval using conceptual structures. In *Proceedings of IEEE Conference on Intelligence Information Processing ICIPS '98*, pages 217–221. IEEE Press, 1998.
13. G. Ellis. *Managing Complex Objects*. PhD thesis, Computer Science Department, The University of Queensland, 4072, Queensland, Australia, 1995.
14. B. Ganter and R. Wille. *Formal Concept Analysis: Logical Foundations*. Springer Verlag, 1999.
15. B. Groh and P. Eklund. Algorithms for creating relational power context families from conceptual graphs. In *Proceedings 7th International Conference on Conceptual Structures*, LNAI 1640, pages 389–400. Springer Verlag, 1999.
16. B. Groh and P.W. Eklund. Dynamic hyper-linking by querying for a FCA-based query system. In *Proceedings of the Forth Australasian Document Computing Symposium*, pages 9–14, 1999.
17. O. Haemmerlé. *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels*. PhD thesis, Montpellier II University, 1995.
18. P. Martin and P. Eklund. Embedding knowledge in Web documents. In *The 8th International World Wide Web Conference, (WWW8)*, pages 324–341. Elsevier, 1999. Also published in *Int. Journal of Computer Telecommunication Networking*, Vol 31, 1403–1419, 1999.
19. P. Martin and P. Eklund. Knowledge indexation and retrieval and the World Wide Web⁹. *Intelligent Systems — Special Issue on Knowledge Management and Knowledge Distribution over the Internet*, May, 2000.
20. P.H. Martin. The WebKB set of tools: a common scheme for shared WWW annotations, shared knowledge bases and information retrieval. In Mary Keeler Leroy Searle Dickson Lukose, Harry Delugach and John Sowa, editors, *Proceedings of the CGTools Workshop at the 5th International Conference on Conceptual Structures ICCS '97*, LNAI 1257, pages 595–588. Springer Verlag, 1997.
21. J. Nanard, M. Nanard, A. Massotte, A. Joubert, H. Betaille, and J. Chauché. Integrating knowledge-based hypertext and database for task-oriented access to documents. In *DEXA '93*, LNAI 720. Springer, 1993.
22. S. Prediger and R. Wille. The lattice of concept graphs of a relationally scaled context. In *Conceptual Structures: Standards and Practices*, volume 1640 of *LNAI*, pages 401–411. Springer, 1999.
23. A. Puder and K. Romer. Generic Trading Service in Telecommunication Platforms. In *Proceedings of the International Conference on Conceptual Structures (ICCS '97)*, LNAI 1257, pages 551–565. Springer, 1997.
24. J. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1983.
25. J. Sowa. Conceptual graphs summary. In L. Gerholz T. Nagle, J. Nagle and P. Eklund, editors, *Conceptual structures: Current theory and practice*, pages 3–52. Ellis Horwood, 1992.
26. R. Wille. Line diagrams of hierarchical concept systems. *International Classification*, 11:77–86, 1984.
27. R. Wille. Conceptual graphs and formal concept analysis. In M. Keeler H. Du-lugach and D. Lukose, editors, *The 4th International Conference on Conceptual Structures*, number 1257 in *Lecture Notes on Computer Science*, Berlin, 1997. Springer.