Knowledge Acquisition using Documents, Conceptual Graphs and a Semantically Structured Dictionary

Philippe MARTIN

INRIA - ACACIA project
2004, route des Lucioles - BP 93
06902 Sophia Antipolis Cedex France
Phone: (33) 93.65.76.45 Fax: (33) 93.65.77.83
E-mail:phmartin@sophia.inria.fr

Abstract

In this paper, we first show how in CGKAT, our knowledge acquisition tool, any document element and its semantics may be represented using the Conceptual Graphs formalism (Sowa, 1984) and a structured document editor. Then, we study the kinds of hypertext links that may be set between documents elements and concepts or relations of the knowledge base (such links enables the use of search techniques on the KB for finding information within the documents). In a second part, we detail the top-level ontologies (for concepts and relations) proposed by CGKAT and its exploitation of a semantically structured dictionary for guiding knowledge representation and easing its later reuse.

1 INTRODUCTION

In the Knowledge Acquisition process, electronic documents may be used at all stages of knowledge acquisition: *some may be expertise sources* (e.g. interview transcriptions and technical reports), *others may be generated* during the Knowledge Based System (KBS) design phase (e.g. documents of specifications, technical documentation and documents given by the KBS for lengthy explanations on its knowledge or on its reasoning).

Information scattered within the documents may detail different aspects of the same knowledge or may detail this knowledge at different levels. Knowledge Acquisition Tools (KATS) may help the knowledge engineer to retrieve information in electronic documents, using the logical structures of the documents, data analysis techniques, or natural language processing techniques. KATs may also enable the knowledge engineer to set hypertext links between document elements and entities of the knowledge base (KB), as for instance in Shelley (Anjewierden and Wielemaker, 1992) and in the K-Station (Albert and Vogel, 1990). Then, the origins of the knowledge of the KB may be retrieved, which is useful for the KB validation and for generating technical documentation. Hypertext links may also be set between the KB and the generated documents, as in MacWeb (Nanard and Nanard, 1993). With the help of these hypertext links between the KB and the documents, the knowledge engineer could ***also use search techniques on the KB*** (e.g. browsers, question-answering systems) ***to find information within the documents***. We have implemented this idea in our KAT called CGKAT.

CGKAT enables its users to represent any document element e.g. a word, a sentence, an image, a paragraph, a chapter and a whole document. Such a function implies 1) a structured

document editor which enables the construction or the selection of document element and the navigation between them; 2) a knowledge representation language which can represent any document element as an entity and the semantic of that entity at the needed level of detail. CGKAT uses the structured document editor Grif (Quint and Vatton, 1992) and the Conceptual Graphs formalism (Sowa, 1984). This formalism has a direct mapping to natural language and enables to represent any word and any statement (e.g. a sentence, an image, a chapter, a document) by a *concept*. Thus, semantic relations between document elements may be represented by *conceptual relations* between concepts. The knowledge modelling methods of current KATs usually do not enable to represent a document element as an entity, and enable to represent only some predefined aspects of information in document elements; then 1) semantic relations between elements cannot be represented; 2) elements can only serve as hypertext anchors for the KB entities. Our tool enables both *representation of document elements and of their relations*, and *the use of document elements as anchors for KB entities*. When a document element is represented by a concept, it becomes an hypertext anchor for this concept but some semantic constraints must be verified by the concept, and there is a semantic link between the concept and the document element. Such semantic links may then be used e.g. for documentation generation. The semantic link is not explicit when a simple anchoring is used. In the first part of this paper, we will examine the respective interest of these two kinds of association for the extraction and the representation of document knowledge, and a method we propose to conciliate them.

Automatic document knowledge extraction provides an help to the knowledge engineer. However s/he still has a lot of work to do for building the KB: s/he has to filter, to interpret and to represent document information according to various goals, common sense knowledge and knowledge acquisition models. At the present time, our tool does not include any automatic knowledge extraction module but *helps* to search information in texts and *to represent this information*. CGKAT proposes to the user a top-level ontology which includes the important high level concept types useful in natural language and knowledge acquisition. The tool exploits the semantically structured dictionnary WordNet (Miller *et al.*, 1990) for proposing concept types (with their known supertypes and subtypes) corresponding to the various known meanings of a lexical entry given by the user. This eases the representation of the documents words and expressions, accelerates and guides the lattice construction, and also facilitates its reuse for other applications.

In the next section, we show the architecture of CGKAT, then the ways the user may build the KB and keep adequate links with documents.

The third section shows functions for exploring, building and using the concept type lattice and the relation type hierarchy. We will discuss their initial content proposed by the tool to the users, and the exploitation of WordNet done by the tool to guide the concept type lattice extension.

Finally, we will compare our work with related ones.

2 BUILDING A KB OF CONCEPTUAL GRAPHS USING DOCUMENTS

After the presentation of the tool architecture, we will see how a document element may be represented in CGKAT and what kinds of link may be set between the document element and the KB entities, and how it helps knowledge modelling.

2.1 The tool architecture

Programming our tool from scratch would have been a tremendous effort. We have been able to reuse or exploit three packages: 1) CoGITo (Carbonneill and Haemmerlé, 1993), a "CG platform for Question/Answer and DataBase capabilities" developed at the LIRMM[1]; 2) Grif (Quint and Vatton, 1992), a structured document editor developed at the INRIA[2]; 3) WordNet (Miller *et al.*, 1990), a public domain on-line lexical reference system developed at Princeton University.

CoGITo is a set of functions for creation, modification, saving and loading a concept type lattice, a relation type hierarchy and CGs (e.g. facts, definitions, schemas and prototypes). The LIRMM team currently implements other packages based on CoGITo that are complementary to the ones of CGKAT, including a question-answering system using query relaxation techniques (Carbonneill and Haemmerlé, 1994) and a cooperative program for the construction of a concept type lattice (Chein and Leclère, 1993). Searches within the CGKAT KB is done with: 1) the LIRMM question-answering system; 2) a browser we presently develop; 3) the menus for handling the concept type lattice and the relation type hierarchy which are presented in section 3. Hence, in this section *we will not detail searches techniques in the KB but their complements for document exploitation, which are links between knowledge and the source or generated documents.*

Grif enables to edit a document structured in various elements (e.g. title, chapter, section, note, paragraph, group of words, figure). The possible orders and presentation of these elements are formally specified in Document Type Definitions (DTDs) and presentation models. Then, the manipulation of these elements (e.g. selection, creation, attribute adding, hypertext connection and presentation change) may be done by the user via the editor, or by programs via a C functional interface called the "Grif Editing Tool Kit". CGKAT exploits this editor and this functional interface in order to offer a graphic interface for the CGs creations, displays and updates: the graphic representation of concepts and relations are typed document elements. Given the DTDs and the presentation models, these graphic representation take little space on disk and may be updated easily. Moreover, hypertext links may be set between them and other document elements (of course, graphic representations always reflect the KB knowledge).

WordNet is an on-line system that exploits a dictionary whose design is inspired by the current psycholinguistic theories of human lexical memory. Some 54,000 word forms of English nouns, verbs, and adjectives are organized into 49,000 synonym sets, each representing one underlying concept (a concept type in the CGs formalism). The synonym sets ar linked either by lexical relations between word forms (e.g. synonymy and antonymy), either by semantic relations between word meanings (e.g. Is-a, Part-of, Cause-of, Attribute and Frame). Thus, a lot of knowledge may be found in WordNet for building a KB for any application. At the present time, our tool only exploits the Is-a relation. For guiding knowledge acquisition and representation, it structures the WordNet high level concept types.

We have built our tool with a client-server architecture. The server is composed by access and manipulation functions on knowledge in the KB and WordNet. The client has methods for document editing and manipulation, and for menu handling. These two parts may either be

1. Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - France.

2. Institut National de Recherche en Informatique et Automatique - France.

compiled together or operate as communicating processes. The following figure shows the architecture. Our contribution is shaded.

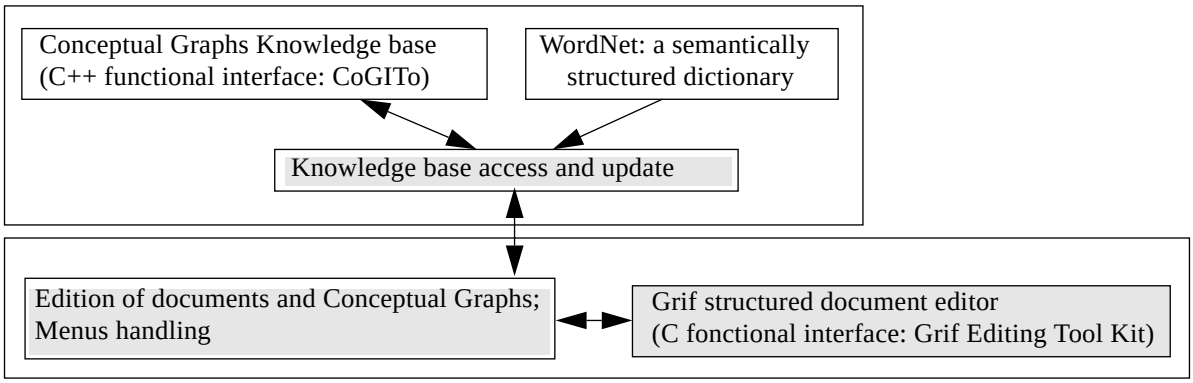

Figure 1: *Our tool architecture* (the greyed parts represent our devcelopments).

2.2 Representation of document element and anchoring of KB entities

Once a document element is selected, it may be represented by a concept or associated to CGs. In order to represent a document element, a knowledge engineer must use a concept and not a CG. Thus, s/he may represent semantic relations between document elements by conceptual relations between concepts. Let us now see which kind of concept to use. According to Sowa (1992), in the Conceptual Graphs formalism, a sentence refering to a situation may be represented using concepts of type Situation[1] and Proposition, as shown in the following extract.

> The next graph describes a situation of a cat on a mat:
> [Situation]->(Dscr)->[Proposition]->(Stmt)->[Graph: [Cat]->(On)->[Mat]].
>
> ... The following graph represents the sentence *There exists a situation described by a proposition stated by a sentence represented by the string "A cat is on a mat"* :
> [Situation]->(Dscr)->[Proposition]->(Stmt)->[Sentence]->(Repr)->[String: "A cat is on a mat"].
>
> There are three possible contractions for simplifying this graph to just a single concept:
> [Situation]->(Dscr)->[Proposition]->(Stmt)->[Sentence: "A cat is on a mat"].
> [Situation]->(Dscr)->[Proposition: "A cat is on a mat"].
> [Situation: "A cat is on a mat"].

We think that any document element which is not a word or a compound word, is a statement of a proposition describing a situation (i.e. a real or imaginary state or process). Such an element may be for instance a group of words, a sentence, a section, an image or a graph. Words might also refer to propositions describing situations but most single words refer to abstract or concrete entities or properties. Since the types of document elements e.g Image, Section or Chapter, are already accessible and handled in the logical structure of a structured document, it would have been a useless complication to represent them in the KB. Hence we decided to represent document elements refering to a situation by concepts of type Proposition[2] and single words by concepts of relevant types (e.g. subtypes of Entity or Property). A document element refering to a situation could not be directly represented by a concept of type Situation, since for example it would be a non-sense to relate such concepts by

1. All states and process are situations and they may be represented by concepts, the types of which are subtypes of State and Process (State and Process are subtypes of Situation).

conceptual relations representing rhetorical relations between document elements e.g. Summary, Elaboration, Restatement, Antithesis and Circumstance. These relation names come from the "Rhetorical Structure Theory" (Mann & Thompson, 1987). Rhetorical relations may be used for hypertext navigation, explanation generation or for studying dialogs between experts.

The knowledge engineer may use arbitrary complex CGs in the referent part of concepts of type Proposition in order to express its interpretation of the content of the document element, at the level of detail s/he needs. When a concept is created for representing a document element, a bidirectional hypertext link is automatically set between the element and the concept.

CGKAT also enables its users to do simple anchoring, that is, to associate with hypertext links, CGs to any document element, or many elements to a CG (a CG may be reduced to a single concept). With that, there is no explicit semantic relation between the CGs and the anchored elements. The user may use simple anchoring to dispatch some information of an element in various CGs which may correspond to various models (e.g. a task model) or modules. We will see in the next section how the user may easily include a same CG or concept in various CGs (or models/modules since they may be represented by CGs).

The user may also set hypertext links between a conceptual relation in a CG and the smallest document elements which led him to set that relation. Thus, even if s/he does simple anchoring, s/he may store the precise origins of the knowledge s/he represents. Additionally, from a semantic point of view, *our tool looks on a document element which is an anchor for a relation, as if it was represented by a concept of type Proposition* with, in its referent part, the graph composed by the relation and the concepts it links. This is useful for the generation of document to answer a request on the KB. Thus, the user may only use simple anchoring if s/he does not need representations of document elements (e.g. to represent relations between them).

The user may also add a CG to the KB without link to any document for example to express some common-sense knowledge. Such a CG, as any CG, may include a concept representing a document element and then may represent semantic relations between elements.

We now detail with two examples how all these links may be set in our tool. We will see in section 3 how the user may choose a type for a concept or a relation.

Figure 2 is a Grif editor window showing an excerpt of a transcription of an expert interview (the expertise domain is car accident analysis/diagnosis (Amergé *et al.*, 1993)). Colored quotes highlight the elements represented by a concept and the elements to which a CG has been associated (single words are not quoted but colored).

Figure 3 is another Grif editor window showing a concept representing the selected document element with very few details. As it is shown, the creation date of a concept and the identifier of its author are automatically registered with each concept. Thus, searches in the KB could be done with some constraints on such information. The source document is not explicitly shown in Figure 3 since every concept which represents a document element is linked to this element by an hypertext link, but this information is also saved in the KB to enable constraints on

2. All types cited here are included in the initial lattice proposed by CGKAT but the type Proposition is the only concept type prescribed by our tool. Actually in our tool this is the only type which may have a CG as a referent. (this is the only "complex" type; Situation is not a complex type). Following Sowa, Proposition is a subtype of Abstract_Entity. Of course, all types may be specialized by the CGKAT users.

source documents during searches. Every user may use a concept for representing a document element and from an element the various concept which represent it may be reached by hypertext navigation: a window like the one in Figure 3 appears. (In the concept shown in this figure, the CG in its referent part only includes concepts with implicit existential referents).

The tool does not allow the user to add in a CG which is in the referent part of a concept representing a document element 'x', of a concept representing a document element which is not included in 'x' (for it is not in the representation of an element that must be represented its semantic relations with other not included elements).

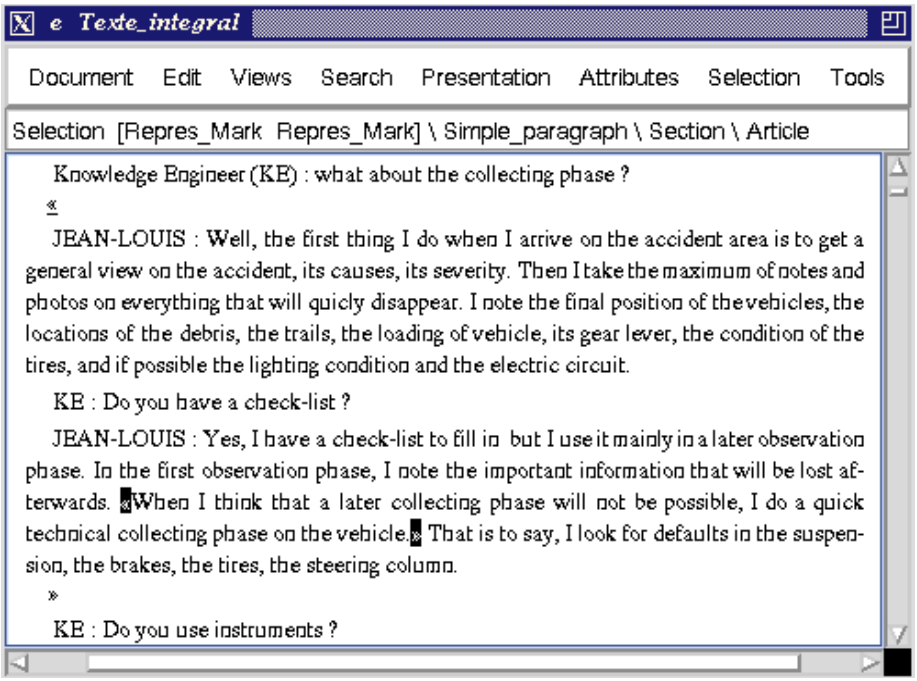

Figure 2: *An expert interview transcription edited with Grif*

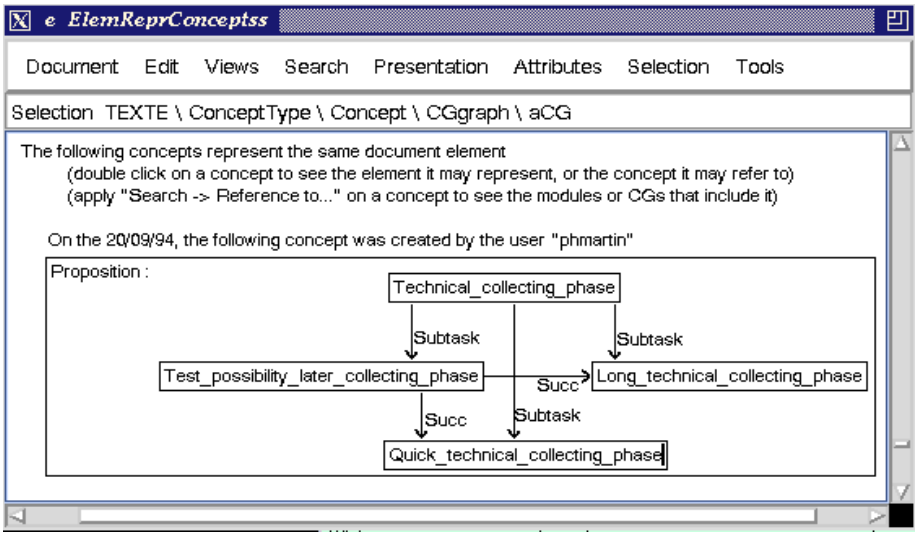

Figure 3: *A representation of the selected document element of Figure 2.*

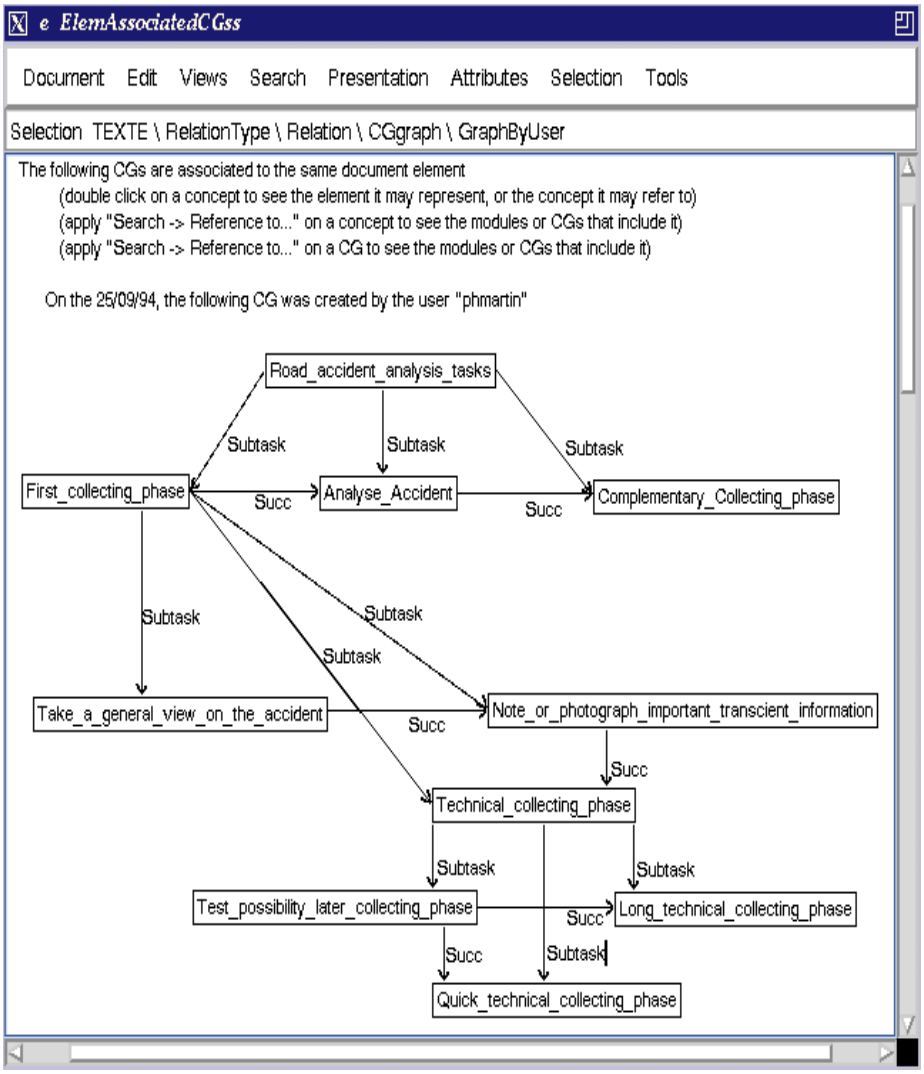

Figure 4: *CG associated to the non selected element of Figure 2.*

In order to speed CG building and hypertext navigation between CGs, the user may very easily add "living copies" of already existing concepts to a CG (a living copy is a copy linked by hypertext link to the original and which is modified when the original is updated; a living copy may be deleted but not modified).

Figure 4 shows a CG that is associated to the non selected element of Figure 2. Each user may associate several CGs to a same element. S/he may include this CG in various modules (or CGs) by the method of the "living copies" (in Figures 3 and 4, the way to access the modules or CGs which do the inclusion, is explained). Thus, the CG in Figure 4 (whose links are of types "Subtask" or "Succ") could be included in a task decomposition tree collecting all the tasks necessary for an application. Hence, the user may focus on a document element and represent different kinds of information it includes, in different CGs, and this way s/he can instantiate various knowledge acquisition models e.g. those of KADS (Wielinga & al, 1992). (Figures 2 and 3 are simple examples; any relation type may be used between concepts, not only Subtask and Succ).

Although information from many documents may then be represented and synthesized in various CGs, the precise origins of this knowledge may be kept: with hypertext links the user may associate to any conceptual relation the smallest document element that leaded to set that relation.

This section showed how the user may build the KB gradually and in a modular way, keeping the exact needed links with the source documents. Two approaches were detailed (Figures 3 and 4). Procedures of translation between the results of these two approaches will be implemented. During document generation, the same kinds of hypertext links could be set. We are now going to see how the user may choose or structure the concept and relation types necessary for buiding the KB.

3 HANDLING AND CONTENT OF THE CONCEPT TYPE LATTICE AND THE RELATION TYPE HIERARCHY

We first show how the user may navigate and handle the concept type lattice, and how our tool exploits WordNet, then we will discuss about the high level structuration of this lattice. Then, we will see a possible structuration for the relation type hierarchy.

3.1 Concept types retrieval and handling

Figure 5 shows with a hierarchically indented list, some[1] subtypes of "Concept", our supertype for all types. By successive selections of concept types, the user may navigate[2] in the lattice (see Figure 6). When the user selects a type, s/he may apply some commands[3] on it. The relation type hierarchy is shown in the same way (see Figure 7). In a future version of the tool, other kinds of lattice or hierarchy will be displayed and handled in the same way: the type of the relations between the handled objects will not be only "Subtype", but could be "Instance", "Subtype&Instance", "Subtype&Entailment", "Succ", "Part", etc. (the hierarchy might have only one level).

3.1.1 Searches of concept types with WordNet

The user may search a concept type by navigation or giving a part of its name. In this last case, the search is done either in the current lattice, or in the WordNet dictionary. We now detail this exploitation of WordNet.

Given a lexical entry e.g. a document word, WordNet extracts its root (the word form) and gives back the word meanings, that is a list of synonym sets (or synsets). Supertypes and subtypes of each synset are also given[4]. We have noticed that for each meaning, it was

1. The depth is controlled by the Max depth" number entry widget and the presence of the keyword "e.g." in the comments. The types which have many direct supertypes are preceded by a ’%’ and the types which only have "Absurd" as subtype are preceded by ’.’. In the future, the "Graphic" button will enable to view the lattice with a graph form.

2. Figure 6 shows the (nine first) supertypes and subtypes of a selected type. When a type has many direct supertypes, each one of them is preceded by a ’#’ and each branch is completely developed (e.g. the supertypes of Physical_Entity).

3. Subtype and supertype adding, deletion, subtypes deletion (the list of the GCs which use these types will be given before the deletion), etc. An "alias" command will soon be implemented and will enable each user to see types under other names; hence the CGs display will be adapted.

4. The results of WordNet are analysed with a simple parser.

possible to build a distinct name with the concatenation of the names in the synset, hence a distinct concept type name (sometimes the tool also has to concat the first direct supertype name in order to make the concept type name distinct from anyother one). Let us take an example: for any of the lexical entries in the synset {life form, organism, being, living thing}, WordNet gives back this synset, and the tool builds the following concept type name from it: W_life_form__organism__being__living_thing. This concept type appears in Figure 6 with a comment part that also comes from WordNet. We have placed it under Physical_Entity (in WordNet, this synset has many subtypes but no supertype).

In this article, we did not alias the concept types coming from WordNet (see Figure 5). Figure 6 shows the only concept type (and its supertypes) found by the command "Search for the following concept (type)" for the lexical entry "auto". Its subtypes, according to WordNet, are also shown since the user selected it. When Wordnet gives several synsets for a lexical entry, each corresponding concept type is shown with its supertypes. Concept types retrieved with WordNet are only propositions: until they are accepted by the user, they are preceded by a '~' and they are removed from the lattice as soon as they are not displayed. They are placed under "Concept" if none of their supertypes is already in the lattice. Thus, the concept types retrieved in WordNet are always subtypes of concept types of the lattice. Hence, in this lattice the accepted WordNet concept types may be structured and their organisation may be different from the organization of WordNet synsets, without any bad effect on the tool exploitation of WordNet. To sum up, the 49,000 concept types of this semantically structured dictionary are "virtually" included in the lattice: they are not actually included but the users may see them as if they were included, and if s/he needs some for its application, s/he can include them[1].

The integration and the use of WordNet knowledge will be further discussed in the next subsection but let us note now that the user will always have to correct or complement some discrepancies in this knowledge. For example, a very little number of WordNet concept types are actually individuals (in the CG formalism sense) e.g. W_Johann_Sebastian_Bach [2]. The users should not include these individuals in the concept type lattice. When the relation "Instance" will be exploited by the "Concept (type) hierarchy handling" menu, the user will easily integrate such knowledge in the KB. The other relations which organize the WordNet synsets will be exploited in the same way.

Another problem is: does the inclusion of concept types proposed by WordNet may change a lattice into a structure which is not a lattice ? Although the WordNet ontology is mainly a tree, the answer may be positive. Hence, a verification procedure should be run after each inclusion, or only upon the user's request since the duration of this check is proportional to the cube of the number of types in the lattice. If the WordNet ontology was not virtually included, each checking would take a very long time. Such checks and other helps to build the lattice will be introduced in CGKAT when it will be connected to the "cooperative program for the construction of a concept type lattice" of Chein & Leclère (1993).

---

1. The user may alias them but not rename then otherwise the link with WordNet would be lost and no subtype could be searched any more for these type.

2. More exactly, the authors of WordNet did not distinguish the Subtype relation and the Instance relation that may exist between word meanings; they used an "Hyponym" relation, which definition is: "a concept represented by the synonyms set {x, x', ...} is said to be a hyponym of the concept represented by the synonyms set {y, y', ...} if native speakers of English accept sentences constructed from such frames as *An x is a (kind of) y*".

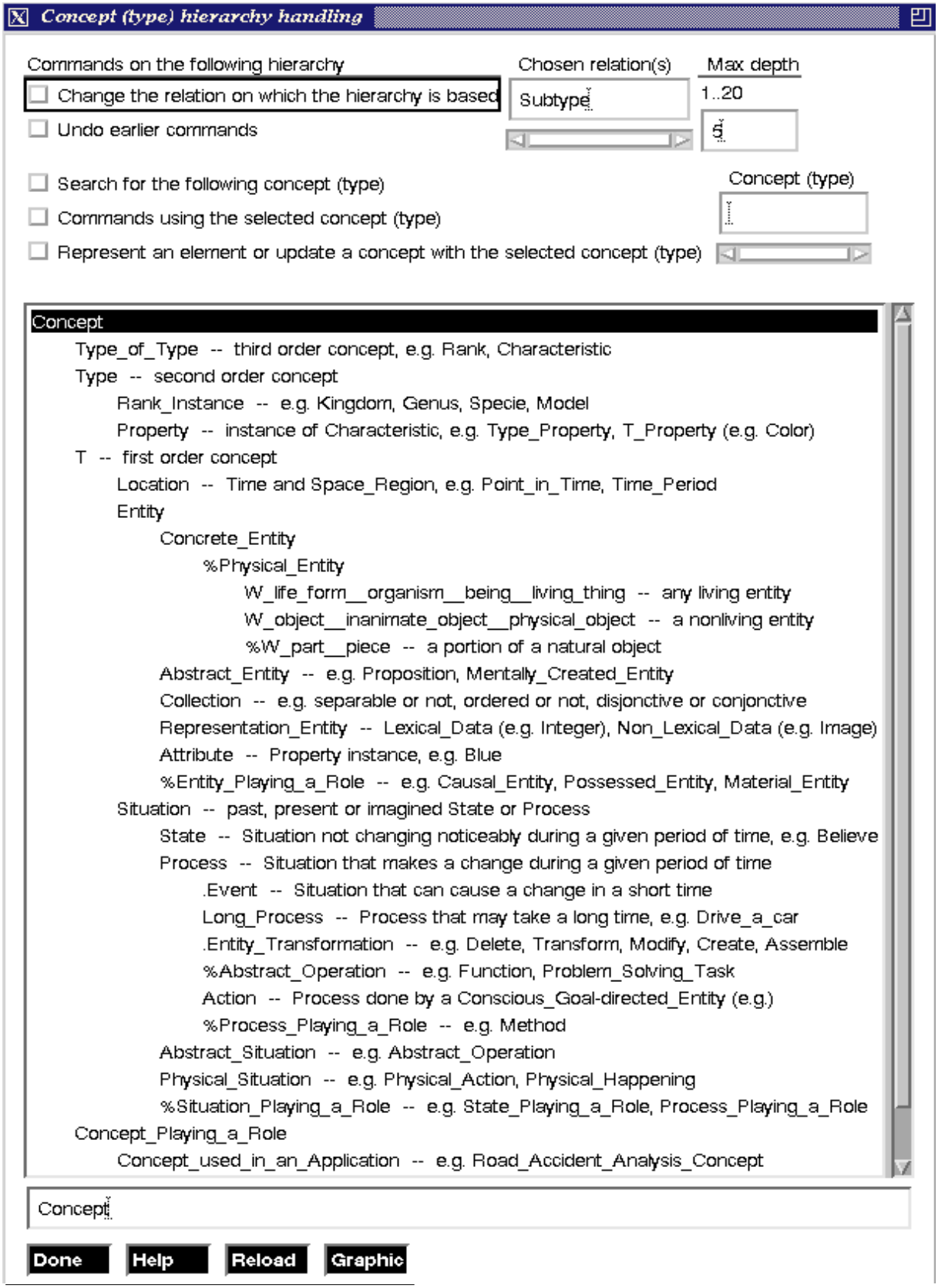

Figure 5: *The concept types handling menu showing some high level types.*

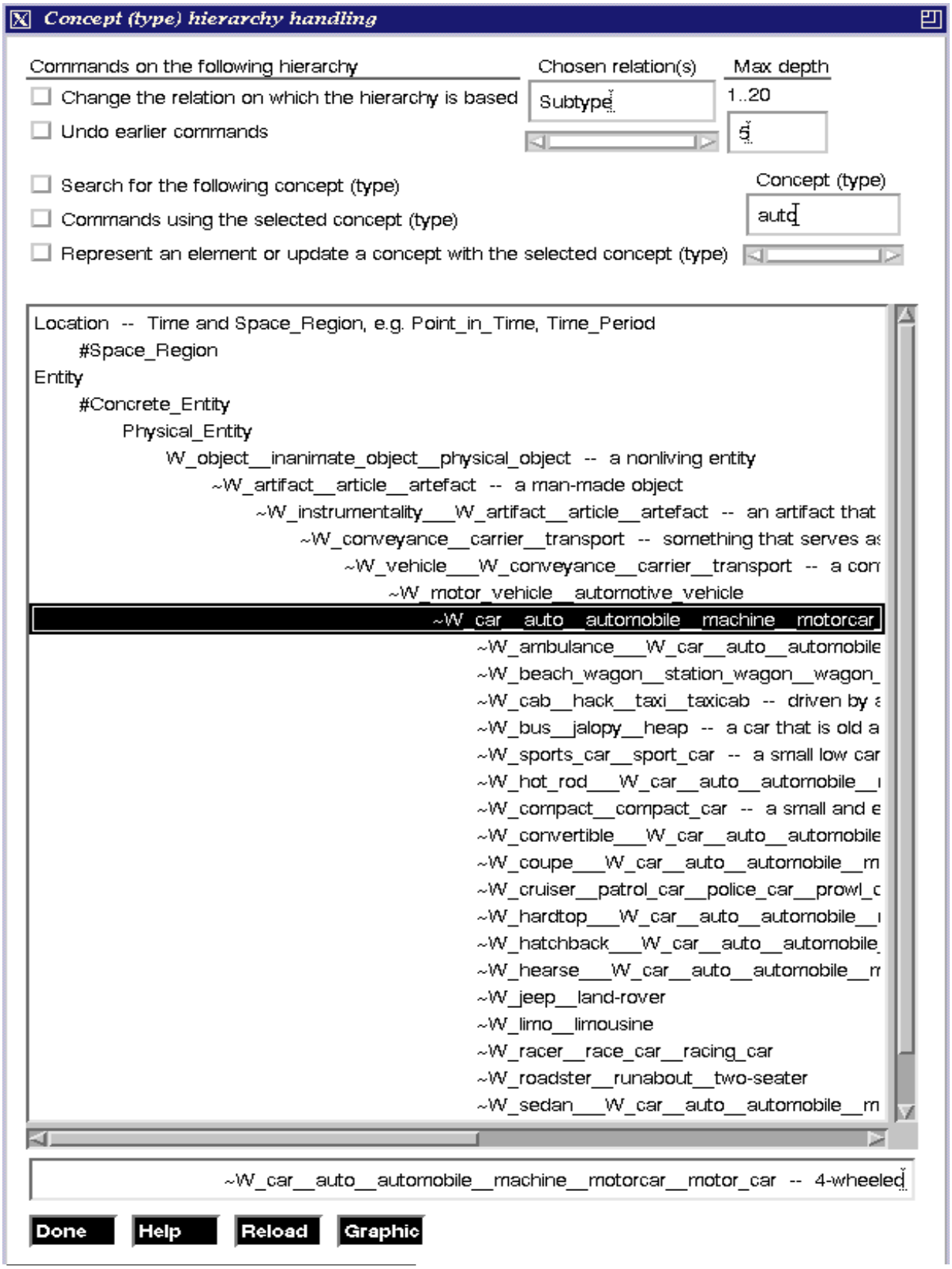

Figure 6: *The unique concept type (with its supertypes and its direct subtypes) proposed by Wordnet for the lexical entry "auto"*

3.2 The concept type lattice

In order to guide and speed the building of the concept type lattice, the tool offers an initial lattice that collects the important high level concept types needed for representing the content of a text or for modelling a KB. The high level concept types of the WordNet dictionary are included in this lattice, then all their subtypes according to WordNet are virtually included.

A KB that is built using concept types coming from WordNet, or specializations of these types, could rather be easily compared with another KB built in the same way since a lot of concept type names used in the two KB would be common as well as their meanings[1]. If those concept types are organized a bit differently in the two BC, automatic procedures could detect the differences and help to solve them[2]. As WordNet is very detailed, the knowledge engineer should rarely have to add intermediate type but rather specialize precise types of WordNet in order to express the shades of meanings needed for his/her application.

Therefore, in order to ease the uses and reuses of the KB knowledge, we suggest to the knowledge engineer to *use and alias or specialize (or alias) concept types coming from WordNet* (for example in the road accident analysis expertise, W_road__route and some structural parts of a road : W_roadbed__roadway and W_shoulder__verge). These types may also be (undirect) subtypes of Concept_used_in_an_application (see Figure 5). Hence, the knowledge engineer may build and work on the minimal hierarchy necessary for its application, without being bothered by the high structuring offered by WordNet and our high level concept types, but without loosing them. *This structuring may be not useful for the final KBS but it is necessary for a good modelling, for powerful searches and inferences, and for easing validation, extension and reuse*. Filters could always be applied when only a part of the ontology is needed.

We now present the high level concept types proposed by the tool. How we specialized these types by the WordNet high level concept types[3] will be detailed in another paper. At present, not all these types are precisely placed.

3.2.1 The upper levels of the lattice

In order to take into account all the aspects of natural language, the top levels of our lattice (see Figure 5) include third order concept types and second order concept types (the user may add higher order concept types).

> Natural languages have *higher order words* whose instances are types rather than individuals. Examples from biology include *kingdom, phylum, class, order, family, genus* and *species* ... and for characteristics, *color, shape, quality,* and *condition.* (Sowa, 1992).

Properties are second order concepts. Attributes are first order concept types and instances of

---

1. The concept types coming from WordNet are precise and have detailed names and comments; therefore we think that they do not induce distant interpretations. This is also an important point for the interpretation and the validation of the BC. Apart from these advantages, if any other large general ontology would exist, it could also be used in the same way by a knowledge engineer for building a detailed and reusable ontology for its application.

2. A lot of problems will have to be solved during the merging of two lattices, for example what to be done if different definitions or schemas are associated to the same concept types ? But this merging, or more generally the reuse of other works, is eased if the works rely on the same basis e.g. WordNet

3. The "WordNet high level concept types" are very poorly structured (its mainly a list). We think that for the use of WordNet in Knowledge Acquisition or Natural Language Parsing, they should first be structured.

subtypes of "Property". As every concept may have properties, and as these properties are generally inherited, we propose that the properties classification follows the concepts classification.

Following Sowa (1992), we divided the first order concepts into entities and situations. Our subtypes for Entity and Situation include the ones proposed by Sowa (1992), and for the collection notions the ones proposed by Pfeiffer and Hartley (1992), and the high level types coming from WordNet. We added a lot of "role types" in order to distinguish them from "natural types". For us, a role type is "a subtype of a natural type, which highlights a role that this natural type can play in some processes: cause, agent, result, etc. (a role type is not a natural type). Here are some examples for entities.

```
Entity_Playing_a_Role --  subtype de Entity
    Causal_Entity -- any entity that causes events to happen (cause of a process)
        Goal-directed_Entity -- Problem Solver or interactional agent Entity
            Conscious_Goal-directed_Entity -- e.g. a person
            Non_Conscious_Goal-directed_Entity -- e.g. an AI agent
        Perhaps_Goal-directed_Entity -- e.g. supernatural forces
        Without_Goal_Entity -- non conscious Entity and not an AI_Agent
    Concept_used_by_a_Process -- e.g. entities of an application
    Recipient_Entity
    W_necessity__essential__requirement__requisite__need -- anything needed
    Representation_Container -- e.g. a text or audio file
    Possessed_Entity -- e.g. a Pet
    Part_Entity -- something determined in relation to something that includes it
    Whole_Entity
    Proposition_Playing_a_Role -- e.g. Belief, Hypothesis, Observations, Norms
```

In order to specialize the Problem_Solving_Task concept type (see Figure 5 at Operation), we began to collect some problem solving tasks in the knowledge acquisition litterature (e.g. Breuker *et al.*, 1987). We represent the models associated to these tasks with concept type definitions and schemas. Thus, a knowledge engineer could use these models in order to collect, abstract and model information from documents.

In order to simplify CGs, some subtypes of Entity or Situation may also be subtypes of Location: for instance, Physical_Entity is also a subtype of Space_Region which is a subtype of Location; thus a relation of type Space_Location may connect two concepts of type Entity or Situation.

3.3 The relation type hierarchy

The functions to display and handle the relation types are similar to the ones used to display and handle the concept types. An additional function, "Types of possible relations for the selected concept (type)", will be described later. (Presently CoGITo and then CGKAT do not enforce the relation type hierarchy to be a lattice; the arguments of every relation type must be defined with known concept types).

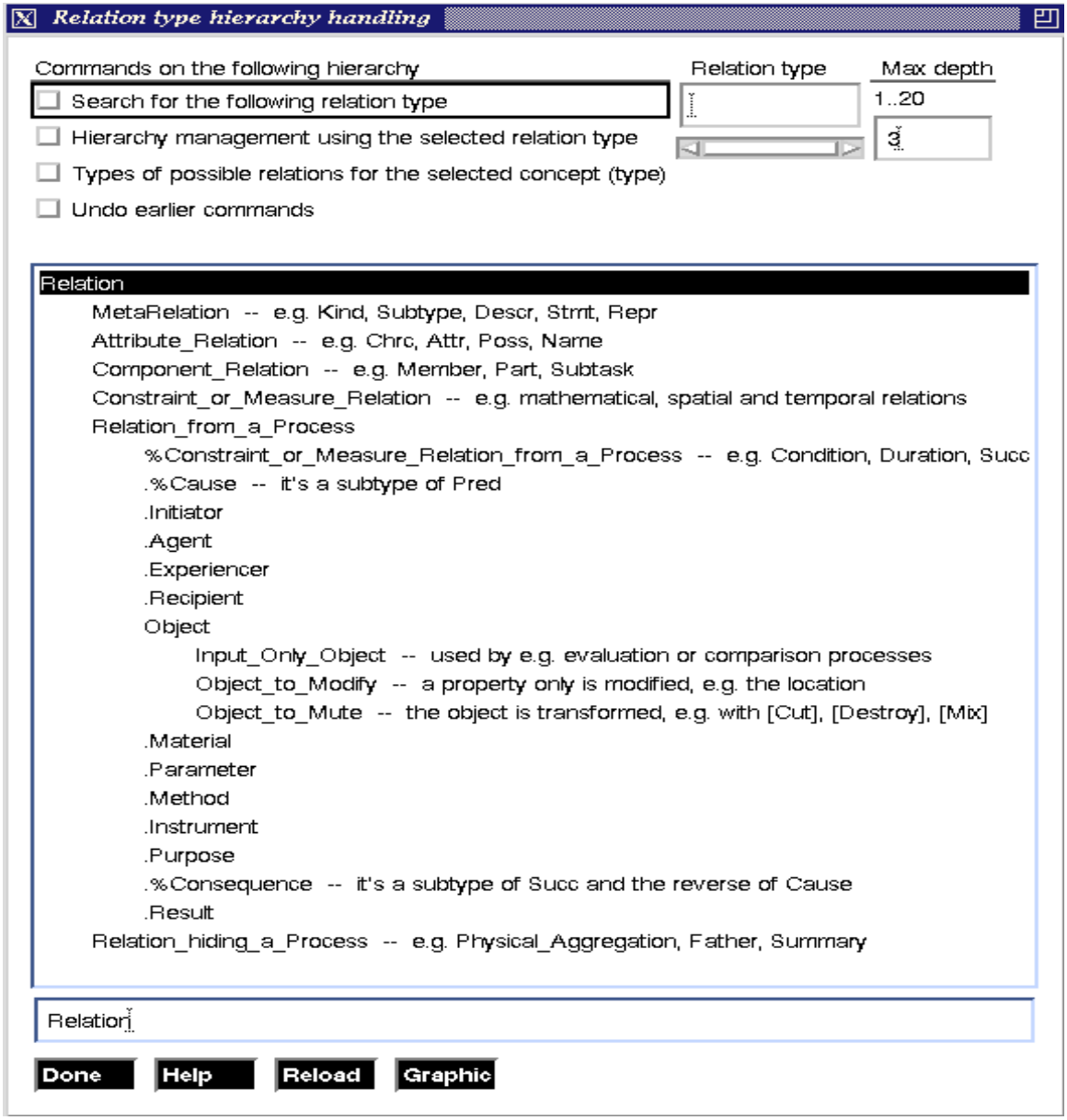

Figure 7: *The relation type handling menu showing high level types.*

Figure 7 shows the high level relation types proposed by the tool. Most of them come from Sowa (1984) and Sowa (1992). We had to precise the "Object" relation in order to specify its different uses in different kinds of actions (see Figure 7). The problem was to find useful high level categories for guiding classification and retrieval of all possible relations (in a knowledge acquisition/representation context).

In order to find those categories, we first noted that relations are often used for expressing the result of a process, or for hiding a process. Here are some examples of such relations.

```
[Bar]->(Chrc)->[Length]->(Meas)->[Meter: 0.25].

(Plus)-                          |  (Physical_Aggregation)-
    <-1-[Number: *]              |      <-1-[Physical_Entity: *]
    <-2-[Number: *]              |      <-2-[Physical_Entity: *]
    -1->[Number: *],             |      -1->[Physical_Entity: *],.

[Proposition: *]->(Summary)->[Proposition:*].
[Proposition: *]->(Contrast)->[Proposition:*].
```

Writing CGs with such relations is shorter than using the corresponding process concept, (e.g. [Summarize] instead of (Summary), with case relations (e.g.Agent, Object and Result)) but we think that there are three drawbacks to do so.

- It does not lead the user to explicit knowledge. Moreover, no search or inference can be made on unexplicit knowledge. For example, compare these two representations.

```
(Divide)-                        |  [Divide]-
    <-1-[Number: *dividend]      |      ->(Dividend)->[Number]
    <-2-[Number: *divisor]       |      ->(Divisor)->[Number]
    -1->[Number: *result]        |      ->(Result)->[Number]
    -2->[Number: *remainder],.   |      ->(Remainder)->[Number],.
```

- If such relations are not defined using their corresponding concept type (like in the following definition example), knowledge is lost for searches and inferences. Conversely, if they are so defined, part of the concept type lattice is duplicated in the relation type hierarchy :

```
relation Summary(x,y) is
    [Proposition: *x]<-(Object)<-[Summarize]->(Result)->[Proposition:*y].
```

- Relations cannot have referent and cannot be precised using relations. For example, it's impossible to specify the author of a summary in a CG that uses the Summary relation type defined above. Thus, unless the user defines a new (non-binary) relation for each new case, or redefine the Summary relation and then update all the CGs that include this relation, s/he will use both concept of type Summarize and Summary relations. This is perhaps not always a conceptual drawback but at least an ergonomic one.

Unary relations are also often used to hide concepts whose referents are continuums of values e.g. the relations Past, Possible and Necessary. Instead we propose to use the relation types Point-in-Time and Modality, and the concept types Time and Modality. Then, knowledge on these notions is uniformly represented, which is good for searches and inferences.

Thus, a distinction must be made between "simple" relations and the ones which hide concepts of type Process. This is why we propose to group these last ones into the categories: "Constraint_or_Measure_Relation" and "Relation_hiding_a_Process". The first category groups relations hiding mathematical functions or measure functions (e.g. spatial and temporal relations). These relations may also be used to express constraints e.g. Before and Equal. The second category groups much more "complex" or "artificial" relations, according to what we have said above. The user may follow the concept type classification for the relations of this category.

The first two categories of our hierarchy, MetaRelation and Attribute_Relation, come from Sowa (1992). The third and the fourth ones are two other useful groupings. The command

"Types of possible relations for the selected concept (type)" lists the relations that could be connected to a concept selected in a CG or to a possible concept of type the one selected in the "Concept (type) hierarchy handling" menu. The list follows the categories order shown in Figure 7; hence, the user can find easily the relation s/he wants.

4 COMPARISON WITH RELATED WORK

Our tool may be used as a knowledge acquisition system but also as a flexible and semantically rich hypertext system[1] or an Information Retrieval (IR) system[2] working on a limited number of documents. The more or less complex KB needed for these three kinds of application, may be built using the next three kinds of complementary techniques (the interest of each one depends on the amaount of information to exploit and the level of detail needed by the application in the representation of the information).

- *Data analysis techniques*: these are statistics on document elements occurrences and co-occurrences in large documents. They may be used in order to make indexes for documents, to extract the terminology of a domain. Thesaurus and grammatical analysis may also be used for better classifications (Bourigault, 1995).
- *Syntactic and semantic parsing techniques*: they enable to extract more or less complex or precise representations of document elements and of their relations. For IR and hypertext, representations may be as simple as SGML tags (or keywords) or as complex as CGs. DR-LINK (Myaeng, 1992) is an IR system currently under development, in which documents are represented by CGs. Myaeng notes that in IR systems, the main criterion for judging the quality of retrieved text is "aboutness", and hence 1) "the IR process is different in nature from knowledge processing for question-anwering", 2) "the limitations of the domain-dependent nature of state-of-the art natural language techniques can be leessened to a great extent". *Knowledge extraction for building a KBS requires much more semantic "understanding"*. At the present time, precise semantic interpretation of text sentences can only be done in very limited domains e.g. in subparts of the medical domain. However, Feng *et al.* (1994) show that the construction of semantic clusters of concepts with the results of a text processing phase on a technical document, is a more accessible task. More exactly, their tools, called TANKA and MaLTe, find equivalent classes of words or phrases, and in order to do so, they exploit two public domain lexical sources, the Collins Dictionary and WordNet (in the same way as in our tool, i.e. by issuing commands and then converting the results into an internal representation).
- *Knowledge is extracted and represented by the user*: actually in KA, even if the user is guided by the results of data analysis or/and natural language processing techniques, s/he always has yet an important workload to do.

---

1. The high and explicit structuring of a KB is not only a good support for hypertext or browsing navigation but also prevents user disorientation as Bernstein (1990) has shown. Besides, the user may use the high-level and precise requests that are possible on a CG database (i.e. the KB of the tool) in order to find information in documents, i.e. for Information Retrieval (IR). However, our tool is not well adapted for document retrieval or large-scale hypertext management, since these applications work on a lot of information that must be extracted automatically or even semi-automatically. An automatic parser could be added to our tool, but then the extracted information would not be complex enough to justify the use of a KB. For knowledge acquisition, very detailed knowledge is extracted and organized, this cannot be done by a natural language parser.

2. IR is a process of identifying all and only those (part of) documents that satisfy user's information needs.

Shelley (Anjewierden and Wielemaker, 1992) and the K-Station (Albert and Vogel, 1989) are KA tools that support respectively KADS-I (Wielinga *et al.*, 1992), a model-driven KA methodology, and KOD (Kuntzmann-Combelles and Vogel, 1988), a more data-driven methodology. These tools enable their users to set hypertext links between portions of text and respectively KADS-I and KOD entities (the text is not structured). The modelling methods supported by these tools are only knowledge acquisition oriented and hence do not enable the user to represent sentences or document elements. On the other hand, CGs may be used to represent KADS-I or KOD entities and their relations. Hence, the user of our tool may follow the KADS-I or KOD methodology, or even both. Representing knowledge with CGs has many advantages: 1) the language has a great representation power, but is easy to use and formal enough to support validation techniques; 2) powerful searches may be done on knowledge using graph matching operations.

Our tool may also be used as a way to set semantically rich hypertext links between document elements. In order to do so, present hypertext system use types for document elements and hypertext relations. Such semantic types are often application-dependent and are hard to find by Natural Language Processing (NLP) techniques. However, MacWeb (Nanard and Nanard, 1993), an hypertext system that is oriented toward KA, semi-automatically extracts a semantic network from a french technical text and then uses it for handling hypertext navigation between various short portions of text (the anchors of the concepts). As this network may be enriched manually, this tool may be used for KA. CGKAT is similar to this one. However, 1) the MacWeb representation language has less representation power than the CG formalism[1]; 2) our tool does not include NLP techniques but let the user filter and represent any document element or the information it contains, at the needed level of detail. Hence, the user may follow any KA methodology (and if s/he wants, s/he may use the results of NLP techniques). Additionally, s/he is guided for the choice and the organisation of his/her concept and relation types.

Although we were not aware of the existence of CODE4 (Skuce and Lethbridge, 1995), the CGKAT features we presented in the second part of this article are similar to some of this general purpose knowledge management system[2], except that CODE4 does not use WordNet but proposes some existing specialized KBs (using the same top-level ontology). However, in the near future, CGKAT will also have many functions to view, compare and handle the knowledge stated by various expert and represented by various knowledge engineers.

1. The CG formalism is logic based and intended to have a smooth mapping with natural language. The MacWeb knowledge representation language is object oriented and apparently does not enable to formally represent quantification nor sets, hence it couldnot formally represent a sentence like "There is only one person to which these two boys may speak to". And there is no computable subsumption relation between portions of the semantic network.

2. CODE4 features a frame-based representation with a number of inheritance and inferencing modes, a flexible graphic user interface with various graphing facilities, a hypertext mode of browsing, the ability to specify functional computation like in a spreadsheet, an optional simple restricted English-like syntax, and document scanning and lexicon management facilities. CODE4 is written in Smalltalk.

5 CONCLUSION

This paper has presented a tool that helps to build a KB. Hypertext navigation is enabled between knowledge and the document elements where it come from. Hence knowledge and results of searches on the KB may be documented. In the near future, searches will be done using a browser that we are currenly developing and a question-answering system developed by another team. We are now studying how to use this last system for answering common explanation demands like e.g. "In which situation X is used ?" or "Why doing X ?".

For guiding knowledge representation and easing its later reuse, given a lexical entry, the tool proposes various concept types. In order to do this, it exploits the on-line semantic word database WordNet and the Wordnet ontology is virtually included in the lattice. In order to add semantics to this ontology, and ease knowledge retrieval and knowledge deduction, we integrated it into Sowa's top-level ontology. At the present time, only the "Is-a" relation between word meanings is exploited. We will do the same kind of work for other relations e.g. "Part-of" and "Cause-of". Any relation type may be defined with the concept type it underlies. Hence, we mainly focused on concept types. However, we also have proposed top-level relation types for structuring the relation hierarchy, and thence guiding the knowledge engineer.

In this article, we have not developed the multi-user and multi-expertise aspects since their handling is not yet implemented. However a concept type aliasing function is being implemented for enabling users to view knowledge according to their lexical preferences and hence easing knowledge sharing. The identifier of a user or an expert may already be associated to any knowledge of the KB: concept, relation, CG, concept type, etc. Then, adapted CGs matching procedures could filter or compare knowledge (of a single KB) coming from various experts or knowledge engineers. We will test the adequacy of the tool functions with the modelling of a "car accident analysis" expertise which involves the cooperation of several experts and which will be represented by various knowledge engineers.

6 ACKNOWLEDGEMENTS

The author thanks the members of the ACACIA team, and especially Dr Rose Dieng, Dr Olivier Corby and Dr Laurence Alpay for their advices on the redaction of this article and the building of the tool. He is also indebted to his referees for their comments and suggestions. The ACACIA team work under contracts with the "Ministère de l'Enseignement Supérieur et de la Recherche" (contract n. 92 C 0757) and the "Ministère de l'Equipement, des Transports et du Tourisme" (contract n. 93.0003).

7 REFERENCES

Albert P. and Vogel C. (1989). *KOD-STATION un environnement intégré pour le génie cognitif.* In "Génie logiciel et systèmes experts", No19, pp. 28-30, Juin 1989.

Amergé Ch., Corby O., Dieng R., Fleury D., Giboin A., Labidi S. and Lapalut S. (1993). *Acquisition et modélisation des connaissances dans le cadre d'une coopération entre plusieurs experts : Application à un système d'aide à l'analyse de l'accident de la route*. Rapport intermédiaire du "Ministère de l'Enseignement Supérieur et de la Recherche" (contract n. 92 C 0757).

Anjewierden A. and Wielemaker J. (1992). *Shelley - computer-aided knowledge engineering*. In Knowledge Acquisition (1992) 4, pp 109-125.

Bernstein M. (1990). *Hypertext and technical writing*. In Proc. DEXA'90, Int. Conf. on Databases and EXpert

systems Applications, Vienn (Austria), August 1990.

Bourigault D. (1995). *LEXTER, a Terminolgy Extraction Software for Knowledge Acquisition from Texts*. In Proc. of the ninth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'95), Gaines, B.R. Eds, University of Calgary, Banff, Alberta, Canada, February 26-March 3, 1995.

Breuker J., Wielinga B.J., Someren M., Hoog R., Schreiber G., Greef P., Bredeweg B, Wielemaker J. and Billault J. (1987). *Model Driven Knowledge Acquisition: Interpretation Models*. Deliverable A1, Esprit Project 1098 Memo 87, VF project Knowledge Acquisition in formal domains. Breuker J. Eds.

Carbonneill B. and Haemmerlé O. (1993). *Implementing a CG Platform for Question/Answer and DataBase capabilities*. Proceedings of the Second International Workshop on Peirce, Québec, August 1993.

Carbonneill B. and Haemmerlé O. (1994). *ROCK: un système de Question/Réponse fondé sur le formalisme des Graphes Conceptuels*. In Actes du 9ème Congrès Reconnaissance des Formes et Intelligence Artificielle, Paris, Janvier 1994.

Chein M. and Leclère M. (1993). *A cooperative program for the construction of a concept type lattice*. Research report No 93075 of LIRMM, 1993.

Feng C., Copeck T., Szpakowicz and Matwin S. (1994). *Semantic Clustering Acquisition of Partial Ontologies From Public Domain Lexical Sources: First Experiments*. Volume 1 of Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop; Gaines B.R. Eds, University of Calgary, Musen M., Stanford University; Banff, Alberta, Canada; January 30 to February 4, 1994.

Kuntzmann-Combelles A. and Vogel C. (1988). *KOD: a support environnement for cognitive acquisition and management*. In Safety and Reliability Symposium, 1998.

Liddy D.E. and Sung H.M. (1992). *DR_LINK Document Retrieval using Linguistic Knowledge, Project Description*. In Revue ACM: SIGIR Forum Vol 26 no 2, Fall 92 pp 39-43.

Quint V. and Vatton I. (1992). *Hypertext aspects of the Grif structured editor: design and application*. Research Report No 1734 of INRIA, July 1992.

Mann W. and Thompson S. (1987). *Rhetorical Structure Theory: Toward a functional theory of text organization*. In Text, 8, 3, 243-281.

Myaeng S.H. (1992). *Conceptual Graphs as a Framework for Text Retrieval*. Conceptual Structures: current research and practice; editors: Nagle T.E., Nagle J.A., Gerholz L.L., and Eklund P.W.; England , Ellis Horwood Workshops, 1992.

Miller G.A., Beckwith R., Fellbaum C., Gross D. and Miller K. (1990). *Five Papers on WordNet*. CSL Report 43, Cognitive Science Laboratory, Princetown University, July 1990. (These papers and the system are available by anonymous ftp at clarity.princeton.edu, subdirectory 'pub').

Nanard M., Nanard J., Massotte A.M., Chauché J., Djemaa A., Joubert A., Betaille H. (1993). *La métaphore du généraliste: Acquisition et utilisation de la connaissance macroscopique sur une base de documents techniques.* Proc. JAVA'93 (Journees sur l'Acquisition, la Validation et l'apprentissage), 5èmes Journées Acquisition des Connaissances du PRC-GDR-IA du CNRS, Saint-Raphaël, 31 mars - 2 avril 1993.

Nanard J. and Nanard M. (1993). *Should anchors be typed too ? An experiment with MacWeb*. Proc. HTX93, 5th ACM Conf. on Hypertext, ACM Press, Seattle, Nov. 1993.

Pfeiffer H.D. & Hartley R.T. (1992). *The Conceptual Programming Environment, CP*. In Conceptual Structures: current research and practice (editors: Nagle, T.E., Nagle, J.A., Gerholz, L.L., and Eklund, P.W.), England, Ellis Horwood Workshops, 1992.

Skuce D., Lethbridge T. (1995). *CODE4: A Unified System for Managing Conceptual Knowledge*. To appear in Knowledge Acquisition. (See also http://www.csi.uottawa.ca/~ctran/code4.html).

Sowa J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA.

Sowa J.F. (1992). *Conceptual Graphs Summary*. Conceptual Structures: current research and practice (editors: Nagle, T.E., Nagle, J.A., Gerholz, L.L., and Eklund, P.W.), England , Ellis Horwood Workshops, 1992.

Wielinga B., Schreiber G. and Breuker J. (1992). *KADS: a modelling approach to knowledge engineering*. In Knowledge Acquisition (1992) 4, pp 136-145.