OntoSeek: Content-Based Access to the Web

Nicola Guarino, National Research Council, LADSEB-CNR
Claudio Masolo, University of Padova and University Paul Sabatier
Guido Vetere, IBM, Rome Tivoli Labs

PERHAPS YOU'RE AMONG THE many who've entered a search into a Web browser and received pages of links—only some relevant, many not? Dodging this pitfall—barring the way to the Web's wealth of information—requires successful *content matching.*

Current information-retrieval techniques either rely on an encoding process—using a certain perspective or classification scheme—to describe a given item, or perform a full-text analysis, searching for user-specified words. Neither case guarantees content matching, because an encoded description might reflect only part of the content, and the mere occurrence of a word (or even a sentence) does not necessarily reflect the document's content.

For general documents, there doesn't yet seem to be a much better option than some sort of lazy full-text analysis, leaving us to sift through those endless results pages. However, if we narrow the field to a relevant class of information repositories—online yellow pages and product catalogs—content-retrieval techniques based on simple representation capabilities and large linguistic ontologies can be both feasible and crucial. We developed OntoSeek, our information-retrieval system, to target these repositories. In this article, we discuss the special characteristics of online yellow pages and product catalogs, examine lin-

FOR YELLOW PAGES AND PRODUCT CATALOGS, STRUCTURED CONTENT REPRESENTATIONS COUPLED WITH LINGUISTIC ONTOLOGIES CAN INCREASE BOTH RECALL AND PRECISION OF CONTENT-BASED RETRIEVAL. OUR ONTOSEEK SYSTEM ADOPTS A LANGUAGE OF LIMITED EXPRESSIVENESS FOR CONTENT REPRESENTATION AND USES A LARGE ONTOLOGY BASED ON WORDNET FOR CONTENT MATCHING.

guistic ontologies' role in content matching, and present OntoSeek's architecture.

Understanding yellow pages and product catalogs

Online yellow pages locate suppliers based on a generic natural-language (NL) description of their products and services; product catalogs let users select a specific product or service offered by a certain supplier. These repositories' peculiarities, with respect to generic Web documents, can be roughly characterized by four parameters (see Table 1 for their estimated values):

- *vocabulary size*: number of concepts nec-

essary to formalize all descriptions in the repository;
- *description complexity*: average number of concepts for one description;
- *description heterogeneity*: average number of semantic relations in a description with respect to the total number of semantic relations necessary to describe the whole set of descriptions; and
- *query specificity*: frequency of queries that closely reflect the content of the encoded descriptions.

If a repository addresses a huge domain, the vocabulary size is necessarily high; this is true for yellow pages, Web documents, and heterogeneous-product catalogs.

Product catalogs employ a technical and

Table 1. Online yellow pages and product catalogs characteristics compared to generic Web documents. The four resource categories, from left to right, correspond to increasing levels of difficulty of content-based information retrieval. We distinguish between homogenous and heterogeneous product catalogs based on whether they describe single or multiple product lines.

Parameter	Yellow pages	Homogeneous product catalogs	Heterogeneous product catalogs	Web documents
Vocabulary size	Moderate/High	Moderate	High/Very high	Very high
Description complexity	Low	Moderate/High	Moderate/High	Very high
Description heterogeneity	Low	Low	High	Very high
Query specificity	High	Low/Moderate	Low/Moderate	Very low

detailed vocabulary and have a high description complexity due to the presence of a large number of semantic relations and constraints—more so than yellow pages. Despite the high variability of each description's content, description structure is fixed for homogeneous product catalogs, because they use the same semantic relations to form each description. So, description heterogeneity is low for homogenous catalogs, but high for heterogeneous catalogs.

Only yellow pages have high query specificity, because in the other cases, users do not need to know all of a desired product's characteristics in advance. (We refer here to product catalogs for electronic commerce, and not to product data-management systems such as those used for integrated manufacturing.)

Recent progress. Let's examine how current information-retrieval techniques access such resources. According to David Lewis and Karen Sparck Jones,[1] we can distinguish three areas of information retrieval:

Text retrieval. The goal is to find relevant documents in a large collection, in response to a user's query expressed as a sequence of words. Generally, the user does not have a clear idea about the text collection's content, and therefore, a precise semantic match between the user query and the relevant documents is not very important. Current techniques based on word co-occurrence analysis integrated with morphological analysis and word stemming appear to work reasonably well in accessing generic documents.[2] However, for yellow pages and product catalogs, the relative shortness of linguistic descriptions limits this technique's performance. Moreover, although text retrieval's main advantage is that it does not require data encoding, we cannot, in this case, assume that good-quality textual descriptions are already available.

Data retrieval. Here, both the queries and data to be retrieved are encoded by a structured list of words, acting as values of a set of attributes established by the system's designer (for example, Reuben Prieto-Diaz's faceted classification schemes[3]). In some cases, these words belong to a fixed taxonomy. However, large word taxonomies can be hard to design and difficult to maintain. Online yellow pages, where various business activities are organized around a predefined set of commercial categories, use this approach in a very simplified form. The approach also works effectively for product catalogs, allowing a classification based on a fixed feature set. The retrieval quality crucially depends, however, on how a certain business or product fits the predefined structure, and on the user's knowledge of the classification scheme. Thus, for data retrieval, this technique depends on the encoding process's quality (and cost). For yellow pages and product catalogs, we observe that brokers are highly motivated to afford these costs.

Knowledge retrieval. For these systems, the query and data-encoding language is much more expressive. This results in increased precision, because the user can represent accurately the data's content structure and formulate sophisticated queries. If the formalism adopted is powerful enough (such as description logics, for instance), we can also exploit a dynamic classification mechanism—freeing the user from a rigid taxonomy. We can then form and match arbitrary descriptions on the basis of an ontology of primitive concepts and relations. However, this forces users to adopt a language that could be too expressive for their purposes. In fact, as Hafedh Mili and his colleagues underline,[4] the expressiveness of the language adopted for data encoding "is limited by the developer's willingness to formulate long and precise queries." If typical queries are relatively simple, detailed descriptions become useless from the retrieval point of view. This approach might, in principle, work well for yellow pages and product catalogs. However, the necessity of designing an ad hoc ontology of primitive concepts and relations—and the computational problems bound to using sophisticated knowledge-representation languages—might constitute a serious practical drawback. On the other hand, adopting knowledge-based techniques to retrieve arbitrary Web documents today constitutes an open research issue.

The role of linguistic ontologies

In knowledge-retrieval systems, an ontology provides the primitives needed to formulate queries and resource descriptions. Simple ontologies, such as keyword hierarchies, might also benefit text- and data-retrieval techniques. As we shall see, a good ontology can significantly increase both recall and precision. However, especially for yellow pages and product catalogs, three main factors limit the practical adoption of ontologies in information-retrieval systems:

- Data's intrinsic dynamics require a continually updated ontology to keep track of new terms.
- Both the query and encoding processes crucially depend on understanding a rigid set of terms (for example, online product catalogs require the user to adopt a set of previously defined category names).
- The vocabulary size and heterogeneous descriptions require a broad-coverage ontology.

Large linguistic resources (such as WordNet[5]) that cover most ordinary English words and encompass both ontological and lexical information offer a way to overcome these limitations.

WordNet is a linguistic database formed by *synsets*—terms grouped into semantic equivalence sets, each one assigned to a lexical category (noun, verb, adverb, adjective). Each synset represents a particular sense of an English word and is usually expressed as a unique combination of synonymous words.

In general, each word is associated to more than one synset and more than one lexical category; WordNet's interface allows for sense disambiguation by (manually) selecting the appropriate synset and category for a given word. Various kinds of semantic relations are maintained among synsets. Among these, the most relevant to our purposes are hypernymy, hyponymy, and antonymy. The first can be roughly assimilated to the usual subsumption relation, while the last links together opposite or mutually inverse terms such as tall/short or

Table 2. Results of querying our commercial descriptions using a flat list of words.

No.	Query	Descriptions found
1	Automobile	1, 2
2	Automobile Retail	1
3	Car Repair	3
4	Motor Repair	
5	Engine Repair	2
6	Motor Exchange	

Table 3. The commercial descriptions structured around a fixed set of attributes.

No.	Business type	Activity	Object	Market area
1	Store	Retail	Radio	Automobile
	Store	Retail	Stereo	Automobile
2	Workshop	Rebuilding	Engine	Automobile
	Workshop	Repair	Engine	Automobile
	Workshop	Exchange	Engine	Automobile
3	Shop	Retail	Car	
	Shop	Repair	Car	
4	Shop	Retail	Jeep	
	Shop	Repair	Jeep	
5	Workshop	Replacement	Motor	
	Workshop	Mending	Motor	

Table 4. The same queries of Table 2, reformulated taking Table 3's structure into account. Precision has increased.

No.	Query				Descriptions found
	Business type	Activity	Object	Market area	
1	—	—	Automobile	—	
2	—	Retail	Automobile	—	
3	—	Repair	Car	—	3
4	—	Repair	Motor	—	
5	—	Repair	Engine	—	2
6	—	Exchange	Motor	—	

4. Jeep Repair and Retail Shop; and
5. Motor Mending and Replacement Workshop.

For our search, we shall compare these encoding and retrieval techniques:

- a flat list of words,
- a structured list of words,
- a flat list of word senses plus the linguistic ontology, and
- a structured list of word senses, using WordNet's ontology.

Table 2 shows the results of adopting a simple word-matching technique using a flat list of words. Assuming that we want to find a place where we can buy an automobile, a query for "automobile" does not succeed at all. The two business descriptions we get do not involve automobile retail, but are rather related to automobile parts, such as radios and engines. Adding "retail" also doesn't help, because it simply reduces the number of wrong answers. "Car repair" correctly returns description 3, but fails to match description 4, which should be subsumed by description 3. "Motor repair" fails to match descriptions 2 and 5, which might be relevant for the search because "motor" is a generalization of "engine," and "mending" is synonymous with "repair." Finally, because of the polysemy of the word "engine," "engine repair" matches description 2 even if we are actually searching for a locomotive repair workshop. In sum, as we've all experienced, both the precision and recall of this method are pretty bad.

Imposing a simple structure on both queries and data encoding can obtain substantial precision increases. For our example, let's assume the attributes appearing in Table 3.

After reordering Table 2's word lists according to Table 3's structure, precision increases (see Table 4) because the bad

child/parent. We see WordNet offering two distinct services: a *vocabulary*, which describes the various word senses, and an *ontology*, which describes the semantic relationships among senses.

Some advantages. Let's examine some examples that demonstrate the advantages of adopting a linguistic ontology coupled with a structured representation to access online yellow pages. We restrict our search to this set of commercial descriptions (adapted from BigYellow, *www.bigyellow.com*):

1. Automobile Radio and Stereo Retail Store;
2. Automobile Engine Rebuilding, Repair, and Exchange Workshop;
3. Car Repair and Retail Shop;

Table 5. The commercial descriptions disambiguated in terms of WordNet senses. Each sense is expressed as a WordNet synset.

No.	Disambiguated description
1	[car, auto, automobile, machine, motorcar], [radio receiver, receiving set, radio set, radio, tuner, wireless], [stereo, stereo system, stereophonic system], [retail, sell retail], [shop, store]
2	[car, auto, automobile, machine, motorcar], [engine], [rebuilding], [repair, fix, fixing, mending, reparation], [substitution, exchange], [workshop, shop]
3	[car, auto, automobile, machine, motorcar], [repair, fix, fixing, mending, reparation], [retail, sell retail], [shop, store]
4	[jeep, landrover], [repair, fix, fixing, mending, reparation], [retail, sell retail], [shop, store]
5	[motor], [repair, fix, fixing, mending, reparation], [replacement, replacing], [workshop, shop]

answers to queries 1 and 2 disappear. Recall remains the same because we have not eliminated the semantic-match problems mentioned earlier.

The retrieval quality improves considerably if we adopt a linguistic ontology such as WordNet. For example, let's add WordNet to a simple matching mechanism, without taking sentence structure into account. This lets us move from a flat list of words to a flat list of word senses for both the queries and resource descriptions. We assume that these senses result from an interactive disambiguation phase where the end user or the data encoder selects the intended sense of each word introduced. Table 5 shows the lists of senses corresponding to our original list of commercial descriptions.

If we reformulate Table 2's word lists in terms of word senses—adopting sense matching instead of word matching—recall is much higher, as illustrated in Table 6. Consider the first query: description 3 is included in the result list because WordNet considers "car" and "automobile" as synonymous. We include description 4 because the synset [car, auto, automobile, machine, motorcar] subsumes in WordNet the synset [jeep, landrover]. In previous methods, query 4 resulted in an empty list. Now, it correctly returns descriptions 2 and 5. We obtain description 2 because "motor" subsumes "engine." We obtain description 5 because WordNet recognizes "repair" and "mending" as synonyms. Moreover, description 2 no longer appears in query 5's result, thanks to WordNet's ability to disambiguate terms.

As seen before, we can obtain a further precision increase by adding structure to the representation formalism. For instance, adopting the facets representation in Tables 3 and 4 can improve Table 6's results. This is because we can eliminate description 1 from the results of queries 1 and 2 and eliminate description 2 from queries 1 and 3's results.

We have seen how, at least for online yellow pages, the combined use of linguistic ontologies such as WordNet and structured representation formalisms can help an information-retrieval system to

- decouple the user vocabulary from the data vocabulary, by covering the most common English words;
- increase recall, by exploiting the hierarchy to make generic queries and recognizing synonyms;
- increase precision, through the disambiguation mechanism and the ability to navigate the hierarchy to select specific queries; and
- further increase precision, by considering the structure of queries and descriptions.

OntoSeek

We developed OntoSeek's first prototype as a result of a two-year cooperation between Corinto (*Co*nsorzio di *Ri*cerca *N*azionale *T*ecnologia *O*ggetti—National Research Consortium for Object Technology), a partnership of IBM Semea, Apple Italia, and Selfin SpA) and Ladseb-CNR (National Research Council-Institute of Systems Science and Biomedical Engineering), as part of a project on retrieval and reuse of object-oriented software components.[6]

OntoSeek is a system designed for content-based information retrieval from online yellow pages and product catalogs. OntoSeek combines an ontology-driven content-matching mechanism with a moderately expressive representation formalism. The following comprises OntoSeek's main design choices:

- The option to use *arbitrary natural-language terms* for accurate resource descriptions in the encoding phase.

- Complete terminological flexibility for the queries, due to a process of *ontology-driven semantic matching* between queries and resource descriptions.
- Interactive assistance on query formulation, generalization, or specialization.
- A state-of-the-art Internet architecture.
- Good recall and precision factors, and reasonable efficiency on massive data volumes.
- Good scalability and portability.

We designed the system to handle both homogeneous- and heterogeneous-product catalogs. Because the latter are more difficult to handle than yellow pages, mainly because of their higher description complexity and heterogeneity, we adopted simple conceptual graphs (see Figure 1) to represent queries and resource descriptions. Compared with simple attribute-value lists, they are much more flexible and significantly more expressive. However, the expressiveness remains rather moderate with respect to, for instance, current versions of description logics. With conceptual graphs, the problem of content matching reduces to ontology-driven graph matching, where individual nodes and arcs match if the ontology indicates that a subsumption relationship holds between them. However, to fully exploit a linguistic ontology, we need to ensure that we can actually link our graphs to it. This means that their labels must be lexical items (forbidding therefore ad hoc expressions such as `has-function` or `part-of`) and that suitable semantic constraints must be introduced. We view such graphs, *lexical conceptual graphs* (LCGs), as simplified variants of John Sowa's conceptual graphs.[7]

When planning our project, we chose to avoid constructing an ontology from scratch and examined existing resources instead. We chose the Sensus ontology,[8] which comprises a simple taxonomic structure (no meaning

Table 6. Table 2's queries disambiguated in terms of WordNet senses. We used the WordNet taxonomy for content matching. Both recall and precision are much higher.

| No.	Disambiguated query	Descriptions found
1	[car, auto, automobile, machine, motorcar]	1, 2, 3, 4
2	[car, auto, automobile, machine, motorcar], [retail, sell retail]	1, 3, 4
3	[car, auto, automobile, machine, motorcar], [repair, fix, fixing, mending, reparation]	2, 3, 4
4	[motor], [repair, fix, fixing, mending, reparation]	2, 5
5	[locomotive, engine, locomotive engine, railway locomotive], [repair, fix, fixing, mending, reparation]	—
6	[motor], [substitution, exchange]	2, 5
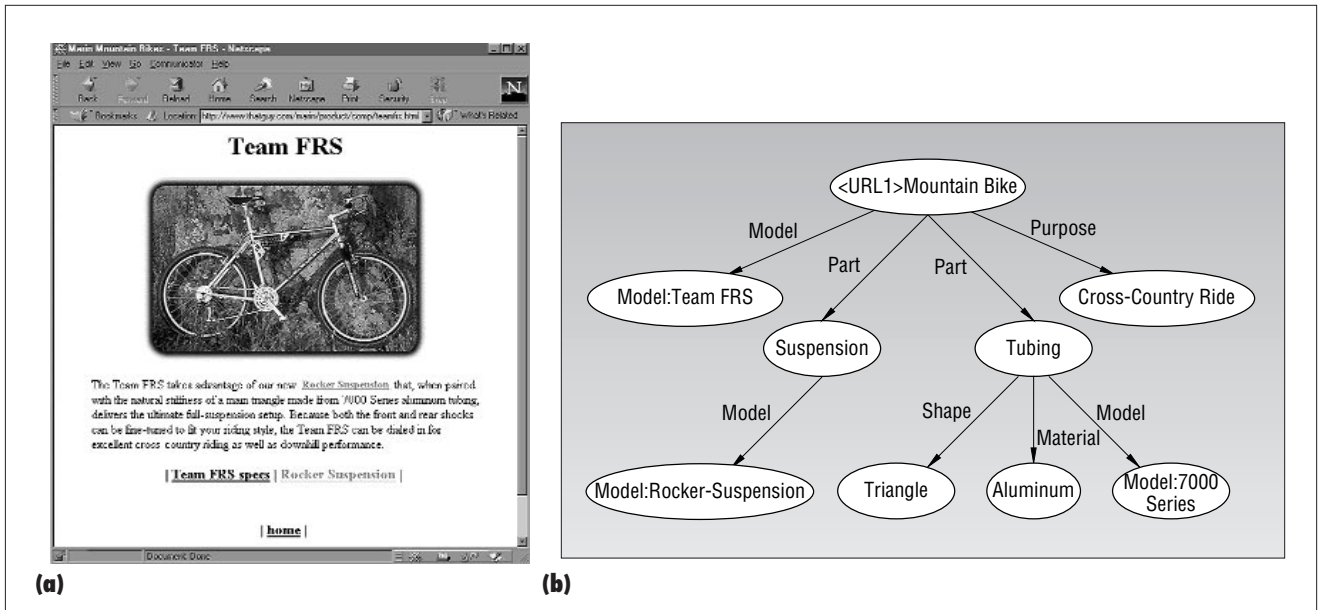
Figure 1. (a) A sample product catalog Web page and (b) its resource description represented as a simple conceptual graph.
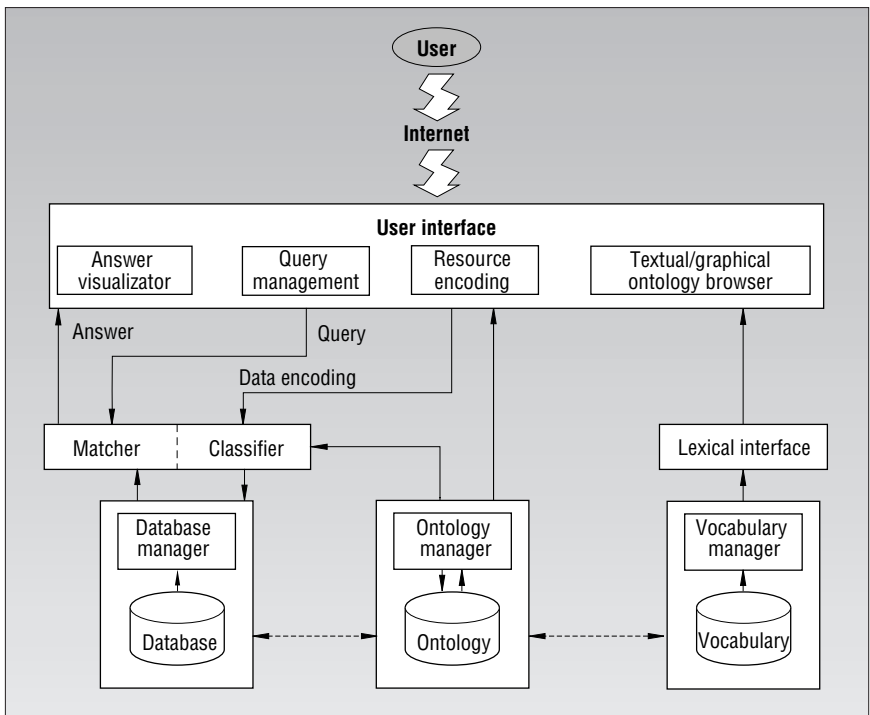

Figure 2. OntoSeek's functional architecture. Solid arrows indicate flow of information; dashed arrows highlight the connections among the three main data structures.

axioms) of about 50,000 nodes, mostly resulting from merging WordNet's thesaurus into the Penman top-level ontology.[9] Despite its strong linguistic orientation and the lack of deep ontological knowledge, Sensus is a very broad ontology endowed with Word-Net's powerful lexical interface, which returns the lexical categories and senses associated with each word. The fact that Sensus was freely available for research purposes also influenced our choice.

Basic architecture

Figure 2 shows OntoSeek's functional architecture. In the encoding phase, a resource description is converted into an LCG with the help of the user interface. The user-labeled nodes and arcs are recognized by the lexical interface, which asks to choose among each word's associated senses, according to the information in the vocabulary. The graph of words is therefore translated into a graph of senses, each one corresponding to a node in the ontology. After semantic validation, performed with the ontology's help, the classifier stores the LCGs in the database. The data-retrieval process works roughly in a dual way. The user represents the query again as an LCG, which undergoes lexical disambiguation and semantic validation. Then, the system searches the database to find the information items described by those graphs that the query subsumes, according to the ontology's taxonomic constraints. OntoSeek then presents the answers to the user as an HTML report.

Resource encoding. Among OntoSeek's key features is the technique used for resource encoding. Basically, it exploits available linguistic resources such as Sensus to enable content matching between graphs using different labels. Most labels currently used in modeling formalisms to denote binary relations (such as `part-of` or `has-function`) do not correspond to lexical entries. In fact, people must often invent ad hoc relation labels when using such formalisms. However, many of these labels are formed from standard nouns, called *relational nouns* by linguists due to their direct relational import. The meaning of such names can be fully understood only in the context of a binary relation: the noun *part* denotes the (unary) property of being a part

(of something not specified), which can be only understood in terms of a binary part-hood relation. Nicola Guarino discussed this situation, defining a relation such as `has-part` as the *relational interpretation* of the noun `part`.[10] Thus, we impose two restrictions on our graphs: first, labels must be correct NL words, guaranteeing a *lexical handle* to interpret their meaning; second, we assume that arcs labeled with nouns always denote such nouns' relational interpretations.

More formally, given an NL vocabulary containing nouns and verbs, we define an LCG as an oriented connected graph satisfying these syntactic constraints:

- Arcs can be labeled only with nouns in the vocabulary (any graph containing an arc labeled with a transitive verb, such as `[<URL1>man]→(love)→[woman]`, can be converted into a basic LCG such as `[<URL1>man]←(agent)←[love]→(patient)→[woman])`.
- Nodes, in general, are labeled with a string of the form `concept[:instance]`, where `concept` can be either a noun or a verb in the vocabulary, and the optional referent `:instance` is an arbitrary identifier.
- For each graph, there is exactly one node called the head of the graph. Such a node is marked with a uniform resource locator prefixed to the label string and enclosed in angle brackets, which identifies the resource that the graph describes. (In the following, if the resource to be described is clear from the context and the graph has exactly one node with no inward arcs, such a node is assumed to be the head and its URL is omitted.)

Before applying to this graph the previous semantic constraints, we must remove polysemy phenomena, by associating the words used for labels to single concepts in the ontology. The OntoSeek system performs this disambiguation phase (see Figure 3) interactively with the help of WordNet's lexical interface. Nodes and arcs are then labeled with a concept identifier, represented by a WordNet synset (for conciseness, we shall assume that the words used as labels in the graphs uniquely identify a certain synset obvious from the context). We can now give a semantic interpretation to the graph:

- Each node labeled with the word "A" denotes a class of instances of the corresponding concept. In the presence of an instance identifier, the node denotes the singleton containing that instance. If A is a verb, the node denotes that verb's nominalization (for example, a node labeled with "love" denotes a class of loving events).
- Each arc labeled C from node A to node B denotes a nonempty relation with domain A and range C ∩ B (see Figure 4).
- As a whole, a graph with head A and URL U denotes a class of instances of A, described by the resource pointed to by U.

The rationale behind the LCGs' lexical and semantic constraints is bound to the choice of exploiting a linguistic ontology to clarify their intended meaning and check their consistency. In fact, LCGs are intended to increase the chances of true content matching:

- Lexical handles let the system use WordNet's lexical knowledge to ask the user (either the end user or the analyst who encodes the data) for possible disambiguation of the words used.
- Restricting arc labels to nouns enforces graphs having a natural intended interpretation, expressible by the following scheme: *If A is linked to B by C, then the [or some] C of A is B.* This linguistic construction (in English) is only possible if C is a noun. For instance, in Figure 4, we can say "the `color` of car c#3272 is red," but not "the `has-color` of c#3272 is red." As William Woods and his colleagues suggested,[11] and as Nicola Guarino discussed,[10] the ability to apply this construct is a necessary condition for considering C as a good attribute for A.
- The constraint on the range of relations permits some degree of semantic check-

Figure 3. The disambiguation process.

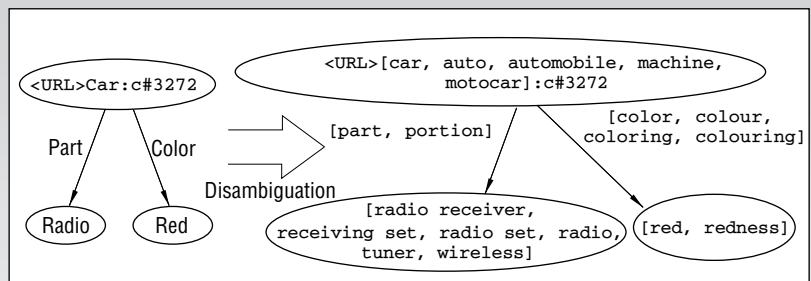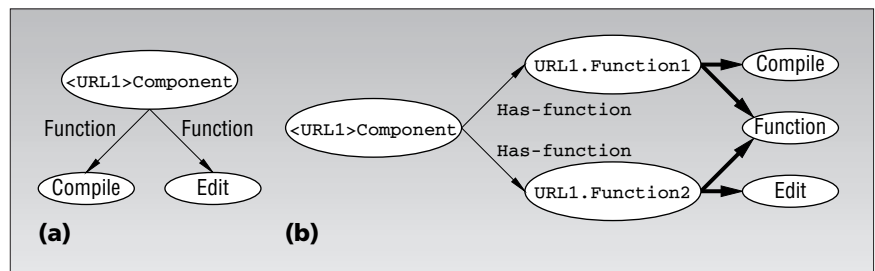| English word	Interpretation	Corresponding synset
Car	Four-wheeled; usually propelled by an internal combustion engine	`[car, auto, automobile, machine, motorcar]`
Color	A visual attribute that results from the light they emit or transmit or reflect	`[color, colour, coloring, colouring]`
Red	A color	`[red, redness]`
Part	Something less than the whole of a human artifact	`[part, portion]`
Radio	An electronic device that detects and demodulates and amplifies transmitted signals	`[radio receiver, receiving set, radio set, radio, tuner, wireless]`

Figure 4. The semantics of lexical conceptual graphs. The LCG in (a) is a shorthand for the semantic network reported in (b). Two different sets of individuals are introduced, representing different sets of values of the `has-function` attribute (represented by the lexical item `function`); the thick arrows denote an ISA relation.
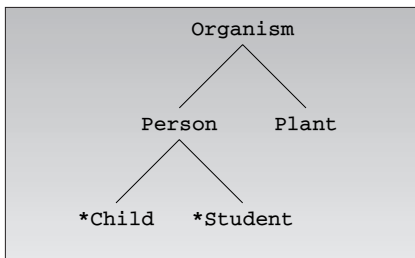
Figure 5. Using the distinction between types and roles to reason about mutual disjointness. Roles are marked with *, and they are assumed as not disjoint (unless explicitly stated otherwise) if directly subsumed by the same type. Types not subsuming each other are assumed to be mutually disjoint.
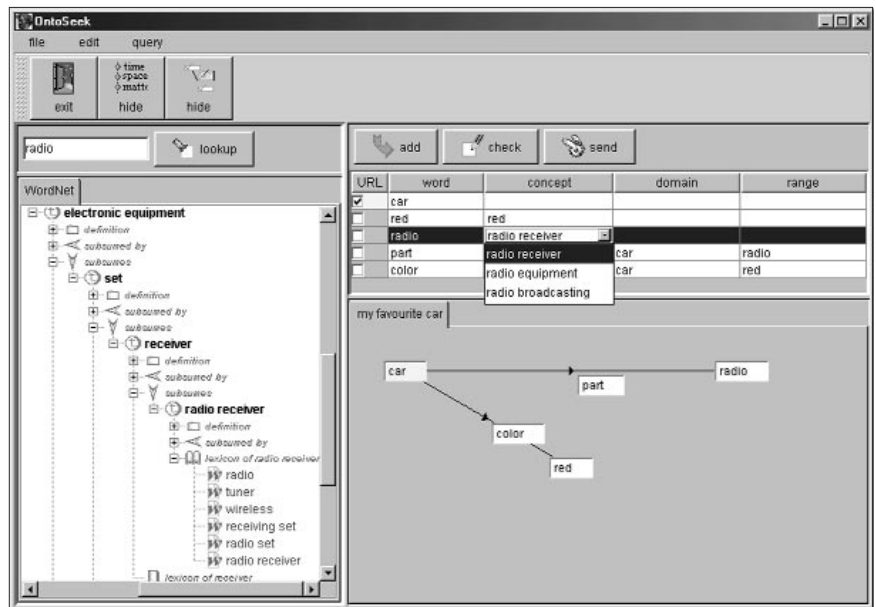

Figure 6. OntoSeek's user interface. The figure shows how the user informs the system that the word "radio" is intended as "equipment" rather than "broadcasting." The nodes chosen as URL selectors are highlighted.

ing, even in those cases (such as the one we have experienced) where the ontology of relations is very poor, and limited knowledge is available about their domain and range. Such a constraint is the simple Attribute Consistency Postulate: *Any X of Y is an X.*[10]

The Attribute Consistency Postulate enables a rough check of LCGs' semantic validity. However, even this simple check might not be easily implemented using an ontology such as Sensus, because it lacks explicit information about disjointness assumptions between nodes. The solution we have adopted comprises distinguishing two kinds of nodes, *types* and *roles*, assuming that

- types that do not subsume each other are always mutually disjoint,
- roles are always subsumed by a type, and
- roles directly subsumed by the same type are assumed as not disjoint (unless explicitly stated otherwise, for example, through an "antonym" link).

Examples of types are `person` and `plant`, while `student` and `child` are examples of roles. Types and roles differ basically in that types are *intrinsically essential* properties—their instances necessarily belong to them.[12] This is not the case for roles because a student can cease to be a student and still remain the same individual.

Suppose now that types and roles are explicitly marked, and you have to check the validity of `[person]→(child)→[student]`. According to the semantics given, we must check whether our ontology admits that a student can be a child—that is, whether `child` and `student` are not disjoint. Now

the type `person` subsumes both `child` and `student`; therefore, we conclude they are not disjoint. Should the user have written `plant` instead of `student`, the system would have concluded that the graph is not valid, because `child` and `plant` are disjoint because `person` subsumes `child`, and `person` and `plant` are disjoint because they are nonsubsuming types (see Figure 5).

Ontology-driven user interface. Figure 6 shows OntoSeek's user interface. A peculiar aspect of OntoSeek is that the ontology not only affects how relevant knowledge is retrieved, but also influences system interaction, at the encoding and end-user levels. The ontology becomes a means of communication between the user and the system. This communication's modality is substantially the same, consisting of the interactive specification of an LCG intended to capture either the resource to be encoded or retrieved. We can distinguish two main phases: node and arc insertion.

Node insertion. Concepts can be either dragged into the graph window from the ontology browser or introduced by the user who enters an English word. The lexical interface analyzes the word and asks the user to disambiguate its meaning according to the various senses that the ontology takes into account (see Figure 3). The two modalities

are, however, highly intertwined, because—after a particular sense has been selected—the user can navigate the ontology seeking possible specializations or generalizations, to refine the description and verify the sense chosen. The ontology becomes a learning tool, useful to master the technical domain, and to express the resource description or the query at the highest level of detail.

Arc insertion. Arcs are entered by selecting a node and drawing an arrow pointing to another node. As mentioned earlier, two modalities are allowed for arc labeling:

- *User initiative.* The user labels arcs, either by directly entering a word or by dragging a concept from the ontology browser. This procedure is analogous to that described above for node insertion, with the difference that here the system checks the LCG's consistency.
- *System initiative.* A pop-up menu asks the user to select a relation among those satisfying the constraints imposed by the particular nodes connected. This choice requires a rich ontology of relations, including information about their domain and range (the so-called *selectional constraints*). Unfortunately, Sensus appears to lack this kind of information, which we manually added for a limited set of concepts just to test our prototype.

Figure 7. The query graph in (a) matches the graph in (b).

Query management. The semantics and process of query-graph construction are reasonably similar to those of encoding graphs, with two differences:

- a variable appears in place of the URL, and
- an arbitrary number of nodes can be marked with such a variable URL identifier.

Because any node can be marked with an URL identifier, no special formalism for representing inverse relations is required. For instance, the query [<X>car]→(part)→ [radio] returns the URLs of the documents describing cars with a radio as a part, while the query [car]→(part)→ [<X>radio] returns the URLs of the documents describing radios as part of a car. Moreover, the multiple query [<X>car]→(part)→[<Y>radio] can be composed to search for both kinds of documents. Just to make it clear, we remark that a query graph *Q* matches a resource graph *R* if

- *Q* is isomorphic to a subgraph of *R*,
- the labels of graph *R* are subsumed by the corresponding labels of *Q*, and
- the head of graph *R* corresponds to a node marked with a variable in graph *Q* (see Figure 7).

The last condition is necessary if we want to retrieve documents strictly focusing on the resources marked by the query. We can relax this condition if the user requests a broader result, including those documents where only an indirect reference to such resources is made. For example, in this case, the query [<X>radio] will match the graph in Figure 7b.

Finally, we should remark that, despite the ease of expressing implicit inverse relations by changing the graph's head, there is no way to match lexicalized inverse relations such as Child and Parent, unless specific knowledge about the relationship between them is available. This kind of knowledge is partially encoded in WordNet via the antonym link, as in the case of Child-Parent or Employer-Employee. Unfortunately, such a link is too generic for our purposes, because WordNet also uses it to encode truly opposite relationships such as Son-Daughter and Mother-Father (see Figure 8).
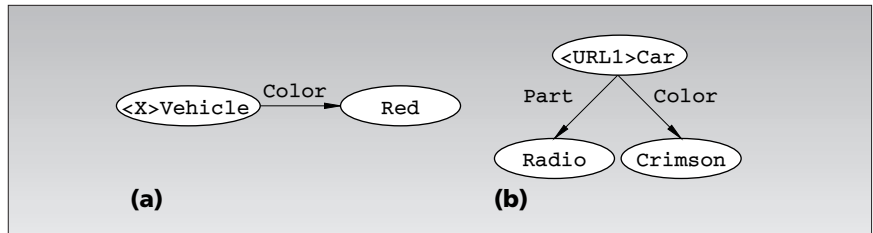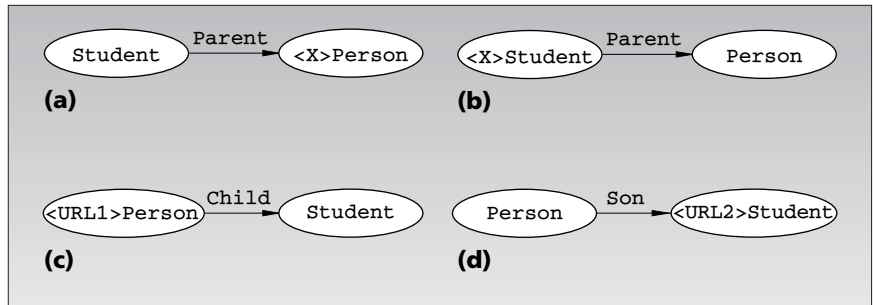

Figure 8. Inverse relations. The queries (a) and (b) can match the descriptions (c) and (d), respectively, only if detailed knowledge about lexicalized inverse relations is available. The WordNet antonym link between Child and Parent can help in the first case, but it does not work in the second case because the antonym of Son is assumed to be Daughter.
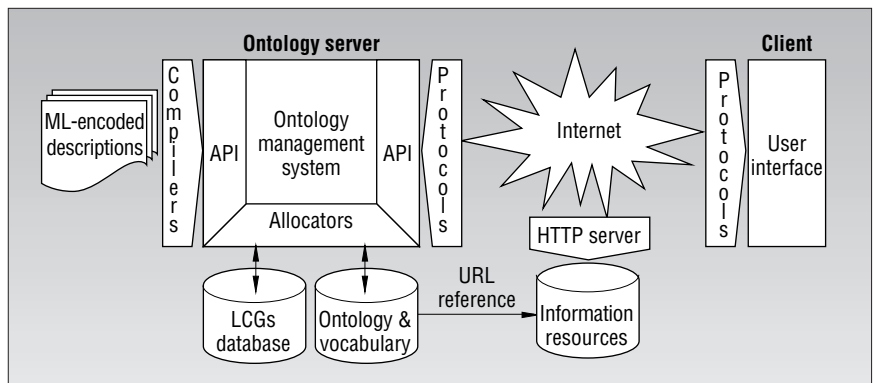

Figure 9. OntoSeek's reference architecture.

Main implementation choices

For our project, we aimed to develop an innovative architecture as well as exploit an effective and industrial-strength technology. To this end, we focused on the following aspects:

- usability,
- low cost,
- good performance,
- openness to different information sources, and
- database support.

The reference architecture in Figure 9 shows how OntoSeek implements the typical client-server paradigm. The architecture's core is an ontology server. The server provides an inter-face for applications willing to access or manipulate an *ontology data model* (a generic graph data structure), and facilities for maintaining a persistent LCG database. End users and resource encoders can access the server through ask/tell communication protocols. The LCG database can also be updated offline by compilers, which accept as input LCGs encoded in Markup Languages (ML), such as HTML extensions or XML.

The minimal expressiveness required to represent the Sensus ontology and the LCGs did not motivate us to adopt a knowledge-representation language based on Lisp or Prolog. On the contrary, we chose to exploit industrial object-oriented languages and technologies, trading off expressiveness and robustness in the most effective way for our
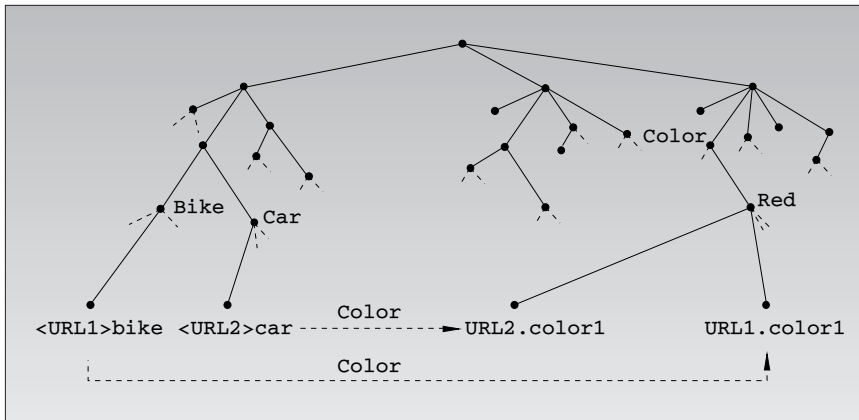
Figure 10. OntoSeek's internal encoding of the two graphs: `[<URL1>bike]`→`(Color)`→`[red]`,
`[<URL2>car]`→`(Color)`→`[red]`.

purposes. Primarily, we wanted to implement a robust and highly standardized C++ framework that encompassed an extensional layer (a repository of LCGs to encode the various descriptions), an intensional layer (the ontology), and the lexical database. We based the implementation on a generic graph data structure, which is further specialized for the two different layers.

The implementation permits permanent storage of the graph data structure on arbitrary memory devices. This is because all allocation, deallocation, and referencing operations are performed through a set of *allocators*—pluggable specialized classes. This means that writing an appropriate set of allocators lets us store the graph on any memory device (for example, a relational database management system), without the need of rewriting parts of the class itself. Thus, OntoSeek can be easily versioned for almost any database management system (DBMS). Moreover, the system can easily integrate legacy databases, provided they comply with the minimum requirements.

We adopted a simple algorithm for the search engine. It's based on a fetch of the resource descriptions appended to the concepts corresponding to the input LCG's node labels, supplemented by a backtracking algorithm to check the relational constraints given by the input-graph structure. Currently, the backtracking algorithm is enhanced by a *first-fail* heuristic (which, roughly, checks less probable links first) to mitigate the NP-completeness of such a computation. Consider, for instance, the two resources URL1 and URL2 in Figure 10. Even if they describe objects with the exact same property, two distinct concepts, marked URL1.color1 and URL2.color2 (corresponding with the "red" nodes in the original LCGs), are appended under the concept acting as filler

("red" in that case) in the ontology graph's extensional layer. For the query `[<X>bike]`→`(color)`→`[red]`, the search algorithm accesses the descriptions appended to `bike` and those appended to `red`, and selects those that the `color` relation actually links together. Notice that LCGs' arcs denote multivalued attributes, so a numbering mechanism is necessary to refer to generic attribute values.

This approach's main advantage, compared, for instance, with the Peirce project to manage conceptual graphs repositories, is that we based OntoSeek's search engine on a classic unification algorithm, rather than on a graph-database partitioning coupled with a sort of homomorphism calculus. This makes OntoSeek's database easier to maintain and make compliant with common DBMS architectures.

As an external language for describing the ontology content, we adopted the basic Ontolingua syntax, by developing an appropriate compiler. (Because Sensus was written in a different language, we translated it into Ontolingua.) We added two slot categories to standard Ontolingua to represent lexical information and resource identifiers in the frame declaration.

The project started up in winter '96, just at the beginning of the Java era. Hence, we had the opportunity to adopt this new and promising technology for developing a powerful Web-integrated user interface. (For more information, see the "Related work" sidebar.) Now, our feeling is that implementing the OntoSeek client side using HTML standard technologies would have hardly been satisfactory. For the client-server protocol, we considered the opportunity to comply with a known specification such as KQML (Knowledge Query and Manipulation Language) or OKBC (Open Knowledge

Base Connectivity), but they were not stable enough at the time of our decision and, in any case, were much too powerful in comparison with our requirements. Another good alternative could have been the development of a specific protocol based on standardized middleware (for example, CORBA), but again, these technologies were still evolving when we made our design decisions, so we eventually opted for crafting an ad hoc solution.

Problems encountered and lessons learned

In terms of the ontology, a first class of problems relates to a lack of ontological information. We found the following major deficiencies in WordNet and Sensus:

Types, roles, and mutual disjointness. As explained earlier, information about mutual disjointness between concepts is crucial to perform even a very rough semantic check, such as that defined for LCGs. Given the unfeasibility of adding explicit disjointness information for each couple of children of a given concept, the solution we adopted, based on a distinction between types and roles, appears to be clean and simple enough. However, it requires a substantial amount of manual work and has only been implemented in a small ontology to test our prototype. Adding this distinction would surely represent a major improvement to WordNet. Consider that most of the original Pangloss nodes in the Sensus ontology are actually types. On the other hand, many of WordNet's topmost synsets are roles, and this generates a confusion in its upper-level structure. This seems to agree with our previous suggestion about the opportunity of constraining the concepts of a top-level ontology to be types.[13] Thus, the utility of a top-level ontology as the main structuring backbone of a larger ontology appears clear.

Selectional constraints. Lack of information about semantic constraints (so-called selectional constraints) on verb arguments (*thematic roles*) makes a more sophisticated semantic LCG check impossible. Therefore, a graph such as `[eat]`→`(patient)`→`[house]` is perfectly valid. What may be worse, `[table]`→`(patient)`→`[house]` is also valid, because WordNet (Sensus as well) gives us no knowledge about the domain of the thematic role `patient`.

Inverse and disjoint roles. We have seen that WordNet encodes the relationship between child and parent or employer and employee in the same way as between son and daughter or father and mother—that is, using antonym links. This seems to be a semantic mistake, because in the latter case we have a disjunction between two concepts and, therefore, a true opposition. In the former case, the two concepts are the domain and the range of the same relation, and they are not necessarily disjoint (somebody can be both an employer and an employee). Of course, this problem regards only roles and could be solved by introducing a new semantic link, with a sure benefit for the usability of Word-Net and similar lexical resources.

A second class of problems regarding the Sensus ontology relates to its structure. Its raison d'être descends from the fact that WordNet's top-level structure is considered to be too poor for knowledge-intensive applications. Penman's top-level distinctions are more structured and seem to allow better control of the common semantic properties of lexical items, being based on the analysis of structural invariants across multiple languages. Therefore, we considered the Sensus ontology a natural choice for the OntoSeek project because it offered a concrete opportunity to perform an experiment of large-scale ontology reuse. However, a number of problems remain.

Relationships between concepts, established based on linguistic criteria, do not often correspond obviously to relationships between classes of entities in the world. Top-level distinctions are hard to understand. Glosses are the only way to understand the meaning of concepts, and there is no way to check their mutual consistency. We have discussed these problems elsewhere[13] and suggested adopting engineering principles based on formal ontology tools to restructure current linguistic ontologies. This would make them more apt to be used for nonlinguistic applications. However, we must underline that, despite these limitations, the large coverage these ontologies offer compensates for the problems caused by their poor structure.

In terms of the representation formalism, we pay the consequences for its poor expressiveness. We have no means to represent negation, variables binding, disjunction, or arbitrary quantifications. In principle, we could adopt a more expressive description logic, adapting to it the lexical constraints used for LCGs. Besides the possibility of a more sophisticated management of queries and data encodings, a further advantage would be the possibility of enriching the ontology with term definitions, providing all the benefits of description-based subsumption and automatic classification. This could allow, for example, matching [parent] with [person]→(child)→[student], just by exploiting the definition of parent. (It seems that roles are more amenable to being defined, while types can generally be characterized only using necessary conditions. A possible step to increase significantly a linguistic ontology's content knowledge could be initially limited to the formal definition of roles based on their glosses. Again, this consideration supports the utility of distinguishing roles from types.) However, we must point out that term definitions were not readily available, and our project aimed to exploit existing lexical resources. Thus, the LCGs' expressiveness seems sufficiently high. Moreover, we reemphasize that the LCGs' power is not in the expressiveness, but in the mix of lexical and semantic constraints imposed on them, which, in our experience, forces users to better understand the nature of their domains of interest.

Finally, we underline that OntoSeek's crucial assumption concerns the fact you must encode the resource descriptions in LCGs to retrieve them. In principle, you can execute such an operation (semi)automatically starting from NL specifications, but in practice it may be impossible for large amounts of legacy data. However, for the particular niche

---

Related work

Recently, a number of proposals for resource-description standards enabling the encoding of descriptions directly into HTML/XML pages have been raised. Aside from considerations regarding these standards' potential utility, we have to stress that our system, in its present version, is based instead on the decoupling of metadata with respect to the resources they describe. Onto-Seek, although the encoding process takes place on the client side, stores the resulting *lexical conceptual graphs* in a database in the server, to support efficient search algorithms. However, the system can be easily adapted to incorporate LCGs encoded in remote documents using markup languages, by developing suitable compilers (see Figure 9 in the main text). In this case, the centralized OntoSeek server could still be used in the encoding phase to guarantee the commitment to the common ontology. Also, the server maintains a global and efficient database of descriptions, which can be automatically updated on request by the remote resources. Thus, resource owners can directly control their own encoding, and the system can take advantage of exploiting markup languages and the HTTP protocol, instead of using data formats and protocols specific to the resource-encoding process.

Currently, the main initiative in this area is the W3C Resource Description Framework. The RDF data model's expressiveness permits relational assertions of the form <subject, predicate, object>, where predicate belongs to a structured vocabulary. *Schemas*, special constructions, provide a formalism to specify a sort of "reference ontology" to be used by a set of homogeneous resources. However, RDF makes no attempt to exploit linguistic tools to enforce semantic consistency. Moreover, a known drawback of RDF, as its authors report, is the difficulty in dealing with huge schemas derived from thesauri (such as Word-Net), because it lacks an efficient way to share them with clients. This problem is currently out from the RDF project's scope.[1]

As the number of encoded Web pages grows, compliance with encoding standards could be a very important enhancement for OntoSeek. However, technologies such as OntoSeek could also be the key to success for encoding standards. Any sufficiently expressive markup-language extension could be easily parsed into LCGs (and vice versa), provided there is a defined way to access a common ontology. Taking the role of ontology server, OntoSeek could produce well-formed resource descriptions for storage as local metadata. Moreover, large amounts of resource descriptions could be stored as LCG databases, making it possible to use linguistic ontologies to search them in a powerful way.

Reference

1. O. Lassila and R.R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Tech. Reports and Publications, www.w3.org/TR/WD-rdf-syntax.

we have in mind, we can reasonably assume that the relative descriptions are moderately simple, so interested companies can themselves encode their product descriptions by accessing the remote OntoSeek server.

An important future application of Onto-Seek, and generally of linguistic ontologies, could be the deployment of multilingual Internet information systems. In fact, it would be possible, at least for the core of Indo-European languages, to identify a common ontological backbone behind the lexical surface of different languages. Currently, a considerable effort is being made for the translation of WordNet to several European languages. (See the EuroWordNet project Web site, *www.let.uva.nl/~ewn*.) Assuming that the L_1 and L_2 languages have been made semantically consistent through a common ontology, the possibility of storing L_1-tagged LCGs and retrieving them with L_2-tagged queries is simply a matter of adding a field indicating the language to which each lexical entry belongs. Stored LCGs are tagged by sense identifiers instead of natural-language entries and, thus, are independent with respect to any particular lexicon. Therefore, users might switch from the L_1 to the L_2 lexical interface (and vice versa) while facing the same database.

L INGUISTIC ONTOLOGIES OFFER immense potential for gathering information resources from the Web. Just taken as they are in their present status (that is, with their poor ontological structure), they can provide substantial improvements to current search systems. Converting them into understandable, clean, and coherent ontologies suitable to drive future information-search systems is not a trivial task, but one well worth addressing. ☐

Acknowledgments

References

1. D.D. Lewis and K. Sparck Jones, "Natural Language Processing for Information Retrieval," *Comm. ACM*, Vol. 39, No. 1, Jan. 1996, pp. 92–101.

2. A. Smeaton, "Information Retrieval: Still Butting Heads with Natural Language Processing?" *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, M.T. Pazienza, ed., Springer-Verlag, Berlin, 1997.

3. R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," *Comm. ACM*, Vol. 34, No. 5, May 1991, pp. 88–97.

4. H. Mili, F. Mili, and A. Mili, "Reusing Software: Issues and Research Directions," *IEEE Trans. Software Eng.*, Vol. 21, No. 6, 1995, pp. 528–560.

5. G.A. Miller, "WORDNET: A Lexical Database for English," *Comm. ACM*, Vol. 2, No. 11, Nov. 1995, pp. 39–41.

6. S. Borgo et al., "Using a Large Linguistic Ontology for Internet-Based Retrieval of Object-Oriented Components," *Proc. 1997 Conf. Software Eng. and Knowledge Eng.*, Knowledge Systems Inst., Skokie, Ill., 1997, pp. 528–534.

7. J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Mass., 1984.

8. K. Knight and S. Luk, "Building a Large Knowledge Base for Machine Translation," *Proc. Amer. Assoc. Artificial Intelligence Conf.* (AAAI-94), AAAI Press, Menlo Park, Calif., 1994, pp. 773–778.

9. J.A. Bateman et al., *A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model*, tech. report, Information Sciences Inst., Univ. of Southern California, Marina del Rey, Calif., 1990.

10. N. Guarino, "Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases," *Data & Knowledge Eng.*, Vol. 8, No. 2, May 1992, pp. 249–261.

11. W.A. Woods, "What's in a Link: Foundations for Semantic Networks," *Representation and Understanding: Studies in Cognitive Science*, D.G. Bobrow and A.M. Collins, eds., Academic Press, San Diego, 1975, pp. 35–82.

12. N. Guarino, M. Carrara, and P. Giaretta, "An Ontology of Meta-Level Categories," *Principles of Knowledge Representation and Reasoning: Proc. Fourth Int'l Conf.* (KR '94), Morgan Kaufmann, San Francisco, Calif., 1994, pp. 270–280.

13. N. Guarino, "Some Ontological Principles for Designing Upper Level Lexical Resources," *Proc. First Int'l Conf. Language Resources and Evaluation*, European Language Resources Assoc., Granada, Spain, 1998, pp. 527–534.

Nicola Guarino is a senior research scientist at the Italian National Research Council's Institute for Systems Theory and Biomedical Engineering. His research interests include ontology design, conceptual modeling, knowledge engineering, and logical modeling of physical objects. He graduated in electrical engineering at the University of Padova. He was chairman of the First International Conference on Formal Ontology in Information Systems (FOIS '98). He is a member of the AAAI, ACM, and Italian Association for Artificial Intelligence (AI*IA). Contact him at the Nat'l Research Council, LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy; guarino@ladseb.pd.cnr.it.

Claudio Masolo is a PhD student in co-tutorship for the Department of Electronics and Computer Science (University of Padova) and the Department of Artificial Intelligence and Cognitive Systems (IRIT-University Paul Sabatier, Toulouse). His research interests include ontological foundations of knowledge representation with special emphasis on axiomatic theories of physical objects and qualitative spatiotemporal representation. He received his degree in electrical engineering at the University of Padova. He is a member of the Italian Association for Artificial Intelligence (AI*IA). Contact him at the Nat'l Research Council, LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy; masolo@ladseb.pd.cnr.it.

Guido Vetere is a staff engineer at IBM Rome Tivoli Laboratories. His research interests range from artificial intelligence and language engineering to object technology. Current work with Nicola Guarino on the OntoSeek project has involved him in work on ontology-driven information systems. He received his degree in philosophy of language, with a thesis on computational linguistics, from Rome University. Contact him at IBM Tivoli Labs, via Sciangai 53, 00144 Roma, Italy; gvetere@tivoli.com.