Logical Information Systems: from Taxonomies to Logics

Sébastien Ferré and Olivier Ridoux

IRISA/Université de Rennes 1
Campus de Beaulieu
35042 Rennes cedex, France
{ferre,ridoux}@irisa.fr

Abstract

Dynamic taxonomies have been proposed as a solution for combining querying and navigation, offering both expressivity and interactivity. Navigation is based on the filtering of a multidimensional taxonomy w.r.t. query answers, which helps users to focus their search. We show that properties that are commonly used only in queries can be integrated in taxonomies, and hence in navigation, by the use of so-called logics. Hand-designed taxonomies and concrete domains (e.g., dates, strings) can be combined so as to form complex taxonomies. For instance, valued attributes can be handled, and different roles between documents and locations can be distinguished. Logical Information Systems (LIS) are characterized by the combination of querying and navigation, and the systematic use of logics.

1. Introduction

Hierarchies are the most common way to organize and retrieve information. We find them in file systems, but also for emails or bookmarks. These hierarchies mimick physical storage structures (e.g., libraries), and hence entail the same problems: documents can be put in only one place, and there must be a strict order between classification criteria. A typical example is the classification of photos. If photos are classified by date, the whole hierarchy has to be browsed in order to collect all photos with some person or of some type (e.g., landscape, animal). Conversely, if photos are classified by their type, it becomes hard to get all the photos of a given period of time. In fact, no single hierarchy is satisfying because different users have different needs, and even a single user has different needs at different times.

Databases allow for the retrieval of documents according to expressive combinations of properties. For instance, it is possible to quickly retrieve all photos of, say, "any of my children in France or Germany, except those before 2000". However, the benefits of navigation are lost as users are not guided in their search, and must know both the query language and the application vocabulary. User queries may lead to an overflow of answers, or no answer at all.

The need to combine querying and navigation has been long recognized, but most proposals are limited.

- For instance, Semantic File System [7] extends the hierarchical model with virtual directories, which correspond to queries. However, no navigation can occur after querying, and queries apply only to intrinsic attributes, not to hierarchy names.

The problem with many approaches is that they use hierarchies as *monodimensional taxonomies*, which means documents can be decribed by only one element of the taxonomy.

- On the contrary, Dynamic Taxonomies (DT) [12, 11, 9] consider *multidimensional taxonomies*, allowing documents to be associated with multiple elements. For instance, a photo can be given a date, a location, and all persons visible on it. Querying is simply selecting documents having some set of taxonomy elements. Navigation is made possible because only the relevant parts of the taxonomy are shown as a summary of the selected documents. Hence querying and navigation can be combined in a same search, in any order.

- Another approach, based on Formal Concept Analysis (FCA) [6], also achieves a tight combination of querying and navigation by automatically deriving the navigation structure from the raw description of documents by sets of attributes [8]. The information retrieval process is similar to DTs, except attributes are not organized in a taxonomy.

In 1999, we started work on Logical Information Systems (LIS) [3, 5], aiming at an even tighter integration of

querying and navigation. Indeed, in the FCA-based approach, document properties can only be boolean attributes, whereas querying often deals with values and patterns (e.g., intervals, substrings). In dynamic taxonomies such properties can be used in document descriptions and queries, but not in taxonomies, which are the basis for navigation. So, if photo dates are represented by a valued attribute, it is not possible to navigate with time periods. Conversely, if dates and time periods are represented in the taxonomy, arbitrary date intervals cannot be used in queries. Of course, both representations could be used together, duplicating the date as a valued attribute and a taxonomy element. However, this could hardly be considered as a *tight* combination; and it would be useful to insert new date intervals from user queries into the taxonomy, for future navigation.

This implies that some of the subsumption relations in the taxonomy are better computed, rather than designed by hand. Regarding these relations as entailments suggests to use logics for defining some part of taxonomies. This use of logics is the key contribution of our logical information systems. Logics support both querying and navigation. For instance, the keywords used in queries will progressively enrich a taxonomy over string-valued attributes (e.g., title, comment). Given a hand-designed taxonomy of locations, various roles between objects and locations (e.g., "in", "from") can be distinguished. This can be done without duplication, so that every change in the taxonomy of locations has to be done only once, because computed and hand-designed taxonomies can be composed into complex taxonomies.

Section 2 first defines what is a *logic*, and how it abstracts over taxonomies. We also present how logics can be composed by *logic functors* so as to form complex taxonomies. Section 3 defines an information base as a *logical context*, and some related operations. Then we describe querying and navigation in LIS, and how they relate to dynamic taxonomies. Section 4 gives practical aspects of LIS.

2. From Taxonomies to Logics

In the context of information retrieval, a *taxonomy* can be defined as a partially ordered set of elements $(E, \leq)$. These elements are used in both object descriptions and queries, and the ordering $\leq$ states subsumption relations between elements. An example is the taxonomy of locations, where elements are places, regions, countries, continents, etc. Reading a place as "somewhere in this place", the subsumption between places corresponds to spatial inclusion. For instance, `Paris` $\leq$ `France`, but not `Paris` $\leq$ `Spain`; and both `France` $\leq$ `Europe` and `Spain` $\leq$ `Europe`.

A taxonomy is generally assumed finite, and with an ordering that is extensionally defined, i.e., designed by hand. However, it is easy to conceive partial orderings that are both infinite and intentionally defined.

- The set of all strings, ordered by the "is contained in" relation: `"The Jungle Book"` $\leq$ `"Jungle"`.
- The set of intervals on integers, ordered by inclusion: $2007 \leq 2000..2010 \leq 2000...$
- The set of intervals on dates, ordered by inclusion: $3\ sep\ 2007 \leq sep\ 2007 \leq sep\ 2007..dec\ 2007$.

In these examples the ordering can be given a mathematical definition, and implemented as a function that takes any 2 elements x and y and returns whether $x \leq y$ holds. With these intensional taxonomies, it is possible to describe a photo with a free comment (string), a size (integer), and a date. Then it is possible to retrieve photos where some word occurs in the comment, that were taken in some period of time, or whose size is in some interval. For navigation, only a finite subset of the elements are made visible (see Section 3), but the insertion of a new element is automatic because its relation to other elements can be computed.

These intentional taxonomies can be seen as *logics* because (1) the elements are properties over objects, and so can be seen as unary predicates, and (2) the ordering is reflexive and transitive, and so can be seen as an entailment. More precisely, this entailment is called *subsumption* (noted $\sqsubseteq$) because it does not apply to statements, but to properties over objects, as in description logics [2].

Definition 1 *A* logic *is a partially ordered set of formulas $(L, \sqsubseteq)$, where formulas are used as descriptors on objects and patterns in queries, and the ordering is the* subsumption *relation between formulas, and can be intentionally defined.*

Each logic is implemented as a module that defines an internal representation, a parser and a printer for user interaction, and a subsumption checker. This definition is compatible with hand-designed taxonomies. The taxonomy of locations is the logic whose formulas are location names, and the subsumption relation is the transitive closure of the taxonomy seen as a graph. To summarize, consider that we have the following logics:

- `Location`: taxonomy of locations, as a logic,
- `String`: logic of strings and substrings,
- `Integer`: logic of integers and intervals,
- `Date`: logic of dates and intervals of dates.

These logics generally correspond to monodimensional taxonomies, while this is not required. For instance, a photo is usually given one location, one date, one size, and one comment. However, we definetely need to combine these logics to form a multidimensional logic, where each object

can be given at the same time a location, a date, a size, and possibly several comments. To this purpose we introduced *logic functors* [4], i.e., functions from logics to logics. They are implemented as parameterized modules, i.e., functions from modules to modules. One of these logic functors is `Sum`, and produces the union of 2 logics. Its effect on taxonomies is to put them side by side. Then a common root is added by applying another logic functor `Top`, whose only effect is to add a top element that subsumes all elements. The logic allowing to describe photos with 4 kinds of properties, as mentioned above, is defined as

```
L1 = Top(Sum(Location,
    Sum(String,Sum(Integer,Date)))).
```

In some applications, locations may be used under different roles. For instance we may want to distinguish between where some person lives *in*, and where she comes *from*. The composite logic `Top(Sum(Location,Location))` is unsatisfying because (1) it duplicates the taxonomy of locations, and (2) elements from the 2 copies cannot be distinguished. Instead we introduce a taxonomy of roles as a logic `Role`, which contains at least the roles `in`, `from`, and a top role `any` meaning "any role". Then we combine this logic and `Location` with the logic functor `Prod`:

```
L2 = Prod(Role,Location).
```

The elements of `L2` are pairs $(role, location)$, and 2 pairs are ordered if the first and second parts are respectively ordered. For instance, a person living in Paris and coming from Spain is described by `in Paris` and `from Spain` (to be read "from somewhere in Spain"). The following subsumption relations are verified:

- `in Spain` $\sqsubseteq$ `in Europe` $\sqsubseteq$ `any Europe`,
- `in Spain` $\sqsubseteq$ `any Spain` $\sqsubseteq$ `any Europe`.

We emphasize the fact that these relations are automatically deduced from the relations between roles on one hand, and between locations on the other hand. If a new location, say Italy, is inserted in the taxonomy of locations, all pairing of a role with Italy become virtually present in `L2`. The same happens for the insertion of a new role. This results in an expressive logic for describing and querying data, while requiring no additional effort from the information system designer.

The logic `L2` can replace the use of `Location` in logic `L1` or, even better, the notion of role can be generalized to all kinds of concrete domain logics, resulting in the logic

```
L3 = Top(Prod(Role,Sum(Location,
    Sum(String,Sum(Integer,Date))))).
```

In this logic the elements of the taxonomy, i.e., properties over objects, can be seen as valued attributes over various concrete domains. Additional concrete domains can then easily be added to this logic definition.

3. Querying and Navigation

Querying and navigating over our logical taxonomies are based on Logical Concept Analysis (LCA), which is a generalization of formal concept analysis that represents properties by logical formulas instead of boolean attributes [5]. In LCA an information base is called a *logical context*.

Definition 2 (logical context) *A* logical context *is a triple $K = (O, L, D)$, where O is a set of objects (e.g., photos), L is a logic equiped with a subsumption relation $\sqsubseteq$ and a most general element $\top$, and D is a mapping from objects to their description, a set of logical properties.*

Object descriptions need only to contain the most specific properties of objects, other properties being implicitely assumed through subsumption. For instance, if a photo is given the property `in France`, it gets implicitely the property `in Europe`. This is taken into account in the definition of *extent*, which defines the set of objects having some property.

Definition 3 (extent) *Given a logical context $K = (O, L, D)$, the* extent *of a property $x \in L$ is defined as the set of all objects having some descriptor subsumed by x:*

$$extent(x) = \{o \in O \mid \exists d \in D(o) : d \sqsubseteq x\}.$$

While this function computes "the objects sharing some property", a dual operation, *intent* computes "the properties shared by some objects".

Definition 4 (intent) *Given a logical context $K = (O, L, D)$, the* intent *of a set of objects E is defined as the set of properties whose extent contains E:*

$$intent(E) = \{x \in L \mid E \subseteq extent(x)\}.$$

The notions of extent and intent are the basis of querying and navigation. We now describe how these operations are defined in our software, CAMELIS. Figure 1 shows a screenshot of CAMELIS user interface, which is made of 3 main components:

- at the top, a *query field* for displaying and editing the current query,
- at the left, a *property tree* displaying a view on the logical taxonomy,
- at the right, an *object list* displaying query answers (here photos).

The contents and relationships between these components are described in the following.

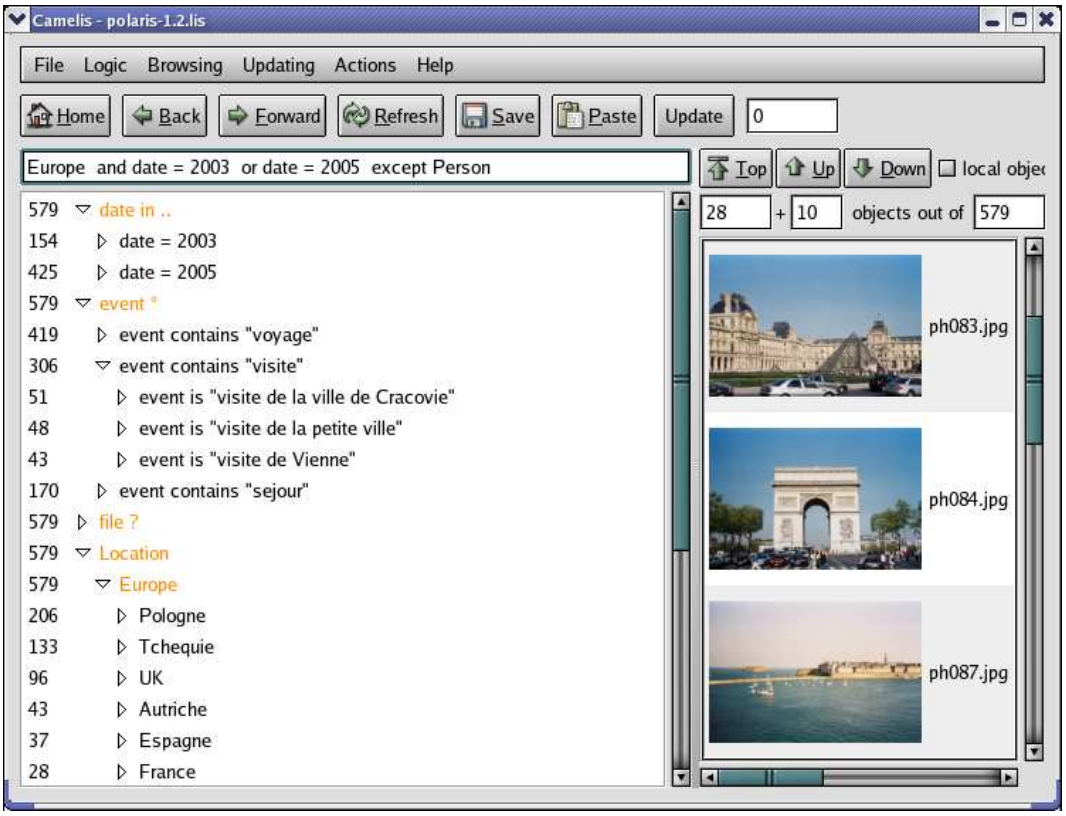

Figure 1. User interface of CAMELIS, exemplified on a photo collection.

3.1 Querying

Queries are arbitrary boolean combinations of logical properties. The definition of extent is extended from properties to complex queries in the natural way (under closed world assumption):

$$\begin{aligned} extent(Q_1 \text{ and } Q_2) &= extent(Q_1) \cap extent(Q_2) \\ extent(Q_1 \text{ or } Q_2) &= extent(Q_1) \cup extent(Q_2) \\ extent(\text{not } Q) &= O \setminus extent(Q) \end{aligned}$$

The object list always displays the extent of the current query. Given a context of photos using the logic `L3` from Section 2, the complex query

```
event contains "journey" and
(in Spain or in Greece) and
not date = 2000
```

retrieves all photos related to some journey taking place somewhere in Spain or in Greece, except those (known to be) taken in 2000. The properties in this query can be used even if they are not visible in the taxonomy. In this case they are automatically inserted and made visible for future navigation. Users also have the possibility to insert new properties, and to modify hand-designed taxonomies, through the user interface.

3.2 Navigation

Navigation in CAMELIS follows the same principles as in dynamic taxonomies [12], but with refinements coming from the use of logic and concept analysis. The property tree displays a subset of the logical taxonomy, showing only properties that are relevant w.r.t. the current query extent.

Definition 5 (support and relevance) *Let $K = (O, L, D)$ be a logical context, and E be the current query extent. The* support *of a property x is defined as $|E \cap extent(x)|$. A property x is* relevant *if its support is strictly positive, or greater than a given threshold.*

In the user interface, the support is displayed at the left of each property. This provides an easy way to read histograms, like the number of photos per year, or per european country. If a property x has a support different to 0 and $|E|$, it can be used to focus the current query Q in 2 opposite ways: Q `and` x, Q `and not` x, making negation accessible through navigation. Disjunction is also ac-

cessible through navigation by multiple selection of properties: after selecting properties $x_1, ..., x_n$, the query becomes Q and $(x_1$ or ... or $x_n)$. These different combinations, and more, which are supported by very few systems, are available through a contextual menu on the property tree.

If a property x has a support equal to $|E|$, this means that $E \subseteq extent(x)$, i.e., x belongs to the intent of the current extent E. In CAMELIS user interface intent properties are distinguished by their text color (grey in Figure 1). Intent properties cannot be used to focus the query, but we use them to widen it. Suppose that after focusing twice we get the query `in France and date = 2007`, and we want to widen it w.r.t. location. The back button would first remove the date property, and editing the query field is no longer navigation. Instead we use the intent properties `in France` and `in Europe`. If `in France` is selected, it is removed from the query, which becomes `date = 2007`. If `in Europe` is selected, it replaces `in France` in the query, which becomes `in Europe and date = 2007`. In both cases the query is made more general, and the corresponding extent is usually larger. In this way a query can be focused and widened w.r.t. any facet (e.g., date, location) in any order.

For efficiency and readability reasons the property tree is initially totally collapsed, and the children of a property are computed on demand, when the user explodes a tree node. This is in contrast with the usual approach in FCA that consists in precomputing the whole navigation structure, the *concept lattice* (exponential in time and space). The computation of children properties is a function of the current extent and the parent property (selected by the user). Several options are available such as minimum support threshold, and whether to sort children by decreasing support or according to some logical order (e.g., chronological for dates, alphabetical for locations).

4. Practical Aspects

CAMELIS is available at `http://www.irisa.fr/LIS/ferre/camelis/`. It has been used in a number of applications: (a) personal photo collection of more than 5,000 photos described by about 50 properties each, taken among 15,000 properties in total (see Figure 1), (b) thousands of music files with automatic extraction of tags (e.g., artist, album) as logical properties, (c) bibliographic files with thousands of references, (d) libraries of Java methods and Objective Caml functions, whose types are expressed in a complex composite logic.

CAMELIS is implemented in a highly modular way so that it can be quickly adapted to various applications. LIS are also implemented as a genuine Linux file system, LISFS [10]. It is therefore visible through existing applications (e.g., text editors, compilers). It has been applied to the management of a whole homedir, and is the basis of a geographical information system, GEOLIS [1].

5. Conclusion

We presented Logical Information Systems (LIS). They offer a combination of querying and navigation that is similar to dynamic taxonomies but integrates complex properties such as dates, intervals and strings in navigation, rather than in querying only. This is achieved by defining and composing logics, where a basic logic can be either a classical hand-designed taxonomy or an intentionally defined taxonomy. This makes it possible, for instance, to distinguish different roles w.r.t. locations, and to automatically insert query terms in taxonomies for navigation purposes. A widening navigation mechanism has also been added as a complement to focusing navigation.

References

[1] O. Bedel, S. Ferré, O. Ridoux, and E. Quesseveur. GEOLIS: A logical information system for geographical data. In *Int. Conf. Spatial Analysis and GEOmatics (SAGEO)*, 2006.

[2] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, Apr. 1997.

[3] S. Ferré and O. Ridoux. A file system based on concept analysis. In Y. Sagiv, editor, *Int. Conf. Rules and Objects in Databases*, LNCS 1861, pages 1033–1047. Springer, 2000.

[4] S. Ferré and O. Ridoux. A framework for developing embeddable customized logics. In A. Pettorossi, editor, *Int. Work. Logic-based Program Synthesis and Transformation*, LNCS 2372, pages 191–215. Springer, 2002.

[5] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3):383–419, 2004.

[6] B. Ganter and R. Wille. *Formal Concept Analysis — Mathematical Foundations*. Springer, 1999.

[7] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. J. O'Toole. Semantic file systems. In *ACM Symp. Operating Systems Principles*, pages 16–25. ACM SIGOPS, 1991.

[8] R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38(5):747–767, 1993.

[9] M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Yee. Finding the flow in web site search. *Communications of the ACM*, 45(9):42–49, 2002.

[10] Y. Padioleau and O. Ridoux. A logic file system. In *Usenix Annual Technical Conference*, 2003.

[11] G. M. Sacco. Dynamic taxonomy process for browsing and retrieving information in large heterogeneous data bases. US patent 6,763,349-2004, 1998.

[12] G. M. Sacco. Dynamic taxonomies: A model for large information bases. *IEEE Transactions Knowledge and Data Engineering*, 12(3):468–479, 2000.