

CAMELIS: Organizing and Browsing a Personal Photo Collection with a Logical Information System

Sébastien Ferré

▶ **To cite this version:**

Sébastien Ferré. CAMELIS: Organizing and Browsing a Personal Photo Collection with a Logical Information System. Int. Conf. Concept Lattices and Their Applications, 2007, Montpellier, France. pp.112–123. inria-00363603

HAL Id: inria-00363603
https://inria.hal.science/inria-00363603

Submitted on 23 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CAMELIS: Organizing and Browsing a Personal Photo Collection with a Logical Information System

Sébastien Ferré

Irisa/Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
Email: ferre@irisa.fr

Abstract Since the arrival of digital cameras, many people are faced to the challenge of organizing and retrieving the overwhelming flow of photos their life produces. Most people put no metadata on their photos, and we believe this is because existing tools make a very limited use of them. We present a tool, CAMELIS, that offers users with an organization of photos that is dynamically computed from the metadata, making worthwhile the effort to produce it. CAMELIS is designed along the lines of Logical Information Systems (LIS), which are founded on logical concept analysis. Hence, (1) an expressive language can be used to describe photos and query the collection, (2) manual and automatic metadata can be smoothly integrated, and (3) expressive querying and flexible navigation can be mixed in a same search and in any order. This presentation is illustrated by experiences on a real collection of more than 5000 photos.

1 Introduction

Formal Concept Analysis (FCA) has been recognized as a good paradigm for information retrieval [GMA93,ML02] because it makes it possible to tightly combine querying and navigation in a same search. Querying alone is not satisfying because it requires from users to know the query language, and to have a precise idea of what they search for. Navigation, by leading users to the result step by step, is more interactive, but the navigation structure is most often very rigid so that only one or a few paths exist to each object (e.g., file hierarchy, hyperlink graph). In FCA, the concept lattice plays the role of the navigation structure. Each concept combines a query as a set of attributes (the intent), and a navigation place as a set of objects (the extent). Attributes can be added to the query in any order, so that a concept can be reached through several paths.

Logical Information Systems (LIS) [FR04] have been introduced (1) to combine querying and navigation, (2) to be reasonably efficient on large collections of objects, (3) to make use of an expressive language for object descriptions and queries, and (4) to be generic w.r.t. the kind of objects and the language. Because of (3) it becomes necessary to do complex reasoning to see whether an object

description match a query (e.g., the object description is a string, and the query is a regular expression). Logics are the right things to encapsulate representation and reasoning facilities. So we defined a generalization of FCA, Logical Concept Analysis (LCA) [FR00], where logical formulas can be used instead of sets of attributes. This makes FCA an instance of LCA, where object descriptions and queries are sets of attributes.

CAMELIS[1] is a complete implementation of a logical information system. It is generic in that a logic module can be plugged in so as to cover different application needs. It uses specific data structures and algorithms so that it is efficient up to 10,000 objects. It has a graphical interface that displays at all time the query that led to the current concept, its extent, and properties to be added or removed from the current query in order to reach neighbour concepts. Both browsing and defining the context are possible through this interface.

Among the various existing applications of CAMELIS, the most convincing is the management of a photo collection. Indeed, photos can be described along many facets like date, location, event, visible persons, visible objects, etc. A file hierarchy enforces a strict order between these facets, making some search hardly possible. Tag-based systems like FLICKR are limited because a photo tagged with 'Sydney' as location will not be answer to a query containing 'Australia'; and a photo tagged with 'formal concept analysis' will not be answer to '...concept analysis'. These limitations are easily solved by dedicated logics: here, a taxonomy of locations, and a logic of string patterns. In this paper, we illustrate the capabilities of CAMELIS and LIS on a real context, the personal photo collection of the author. It contains more than 5,000 photos, and has been incrementally defined since 2003, along the arrival of new photos.

Section 2 assumes an existing context, and presents all the facilities provided by CAMELIS to browse and retrieve photos, from querying by formulas and querying by objects, to different kinds of navigation: i.e., downward and upward, backward and forward, and sideward. Section 3 illustrates the incremental definition of the context with the arrival of a new pack of photos. Section 4 compares CAMELIS with related tools.

2 Retrieving Photos

In this section, we assume the existence of a logical context representing a collection of photos and their metadata. The process of building such a context is developed in Section 3. We recall the basics of Logical Concept Analysis (LCA) [FR00,FR04], whose principles are the same as in FCA, except that *logical properties* partially ordered by a *subsumption* relation are manipulated instead of unrelated attributes.

Definition 1 (logical context). *Let $\mathcal{L} = (L, \sqsubseteq)$ be a* logic*, i.e. a set of properties L partially ordered by a* subsumption *relation $\sqsubseteq$. A* logical context *is a*

[1] See `http://www.irisa.fr/LIS/ferre/camelis/`.

triple $K = (\mathcal{O}, \mathcal{L}, D)$, where $\mathcal{O}$ is a set of object identifiers, and D is a mapping from objects to their description as a set of logical properties.

In our application, object identifiers are file paths or URLs to photos, and their logical description is a set of properties such as location, date, event, etc. These properties are either taxonomic terms or valued attributes over various concrete domains such as date intervals or string patterns. The subsumption ordering is fixed for concrete domains, and designed by hand for taxonomies.

- location: `Montpellier` $\sqsubseteq$ `France` $\sqsubseteq$ `'European Union'`
- date: `date = 24 oct 2007` $\sqsubseteq$ `date in sep 2007 .. jun 2008`
- event: `event is "conference CLA"` $\sqsubseteq$ `event contains "conference"`

An object can be in the extent of a property without having it explicitely in its description, through subsumption.

Definition 2 (extent). *Let $K = (\mathcal{O}, \mathcal{L}, D)$ be a logical context, and $p \in \mathcal{L}$ a property. The* extent *of property p is defined by the set of objects whose description entails p:*

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

Hence, object descriptions can be kept small and precise at the same time. For instance, an object described with some date d will be in the extent of any date interval containing d. Here lies the main benefit of logics w.r.t. attributes. A similar benefit can be obtained with conceptual scaling [GW99], but only for finite domains (e.g., locations), and with redundancy in the scaled context [CFRD06].

Logical properties can be combined by boolean operators *and*, *or*, *not* so as to form more complex *queries*. Their extent is defined inductively as follows for every queries $q1, q2$:

$$\begin{aligned} extent(q1 \text{ and } q2) &= extent(q1) \cap extent(q2) \\ extent(q1 \text{ or } q2) &= extent(q1) \cup extent(q2) \\ extent(\text{not } q1) &= \mathcal{O} \setminus extent(q1) \end{aligned}$$

This definition makes every pair $(q, extent(q))$ an inf-semiconcept of a logical context whose logic is extended with boolean operators. These semi-concepts play the role of both query and navigation place in the following, founding the combination of querying and navigation presented in the introduction.

In the next sections we present the many different ways information can be retrieved in Camelis. This is illustrated on a logical context made of 5,480 photos described by an average of 10 properties each, taken among a set of 28,143. This later figure is not the number of *available* properties, which is infinite, but the number of properties *used* so far.

2.1 The Graphical Interface

The graphical interface of Camelis can be seen in Figure 1. At every time, the *current query* and its *extent* are displayed respectively as a text field at the top



Figure1. A screenshot of the graphical interface of CAMELIS.

and as a thumbnail list at the right. The query is editable, and the extent can be scrolled page by page (the page size can be customized). At the left is displayed a set of property trees, where the hierarchical relations match the subsumption relations between logical properties. As a logic can be any partial ordering, not necessarily a tree, the same property can appear several times. This is not a problem as tree nodes are expanded on demand.

These trees provide a feedback about the current extent, and a support for navigation. Each visible property has a *count* (numbers at the left), and a color that depends on the context and the current query. Let K be a logical context, q be the current query, and p be a property. The count of p is the number of objects in the current extent that have p as a property:

$$count(p) = |extent(q) \cap extent(p)|.$$

A property is not shown if its count is 0, because it tells nothing about the current extent. In order to get fewer properties in the property trees it is possible

to set a minimum count on each property to control the way it is expanded. A property is red- or orange-colored if it is shared by all objects in the extent, i.e. if its count is equal to the size of the extent. This means the set of orange/red properties is the intent associated to the current extent, and red properties are those properties also occuring in the query. Because the count of properties depends on the current extent, property trees need to be recomputed each time the query changes. This makes the navigation trees very dynamic and informative. Specific data structures are used to make their computation efficient, i.e. linear in the size of the context (see [FR04,PR05] for data structures and algorithms). The following sections show how these informations can be used to navigate among photos along various directions, and to get feedback about selected extents.

2.2 Navigating Downward in the Concept Lattice

Firstly, suppose I[2] want to find some photos from a trip in Australia for ICFCA'04. I first expand the property `Location`, and find I have photos from Europe (4859), Africa (162), and Australia (148). After selecting the property `Australia`[3]:

- the query becomes `Australia`,
- the property `Australia` becomes red because it is now part of the query, and it is automatically expanded to show sub-locations of Australia,
- the properties `Europe` and `Africa` are no more visible, because no more relevant (count = 0),
- the extent displays the first page of the 148 selected photos.

The sub-location `'New South Wales'` becomes orange, which means this is the only region of Australia where I have taken photos. More precisely I find that I have been mainly in Sydney (105), and in the Blue Mountains (18).

Now I expand the property `Type` and see there are different types of photos: e.g., buildings (29), animals (34), plants (6). I get interested in Australian living things, so I select both `Animal` and `Plant`, which leads me to the refined query `Australia and (Animal or Plant)`, whose extent contains 40 photos. One of these photos is a portrait, which I do not want, so I select the negation of `Portrait` with the help of a contextual menu. This leads me to the new query `Australia and (Animal or Plant) and not Portrait` (39 photos). By expanding more properties, I discover that these photos were taken in February and March 2004 in New South Wales, that 5 photos of 3 different species of marsupials are present: e.g., kangooroo, koala, wallabi.

These 3 navigation steps led to semi-concepts with more and more precise queries, and hence smaller and smaller extents. This is called *downward* navigation, and its principle is to combine with a `and` the current query and a refinement. A refinement can be a single property, a disjunction of properties,

[2] The pronoun I is used to emphasize a personal experience in the use of Camelis.

[3] French words can be seen in screenshots because it is my real photo collection, but english translations are used in the text for better understanding.

or the negation of a property. When a refinement uses only properties that are relevant and not colored, the new semi-concept is always strictly smaller but not empty. This is a big advantage compared to purely querying systems, where it is common to get empty results.

Figure 1 shows the interface obtained after the previous navigation operations. At this stage, I can either browse the 39 photos in CAMELIS, or launch a diaporama in an external application.

2.3 Navigating Backward and Forward in the Query History

Like in web browsers, it is possible to navigate *backward* and *forward* in the query history. This is mainly useful for opening parentheses on the path of a navigation. Imagine that, while browsing the extent at the end of previous section, I find a photo of a koala, and want to look at all other photos of koala. I first select the property `'koala'`, which leads me to a new semi-concept with 2 photos, and then I can move back to the previous semi-concept. When moving back, scrolling positions are remembered, so that I can go on easily in my browsing of Australian animals and plants.

2.4 Navigating Upward in the Concept Lattice

During downward navigation, I sometimes want to remove or generalize some properties in the query so as to get larger extents: this is *upward* navigation. For instance, I realise I have not enough photos of animals and plants. If I want to remove the last refinement, the back button is a simple way to achieve this. But if I want to remove the first refinement `Australia`, I would need to move 3 steps backward, and re-select the last 2 refinements. I could also edit the query by hand, but users usually prefer to navigate rather than editing queries.

Besides, orange and red properties are shared by all extent objects, and so cannot be used for downward navigation. This makes them available for upward navigation. When a red property is selected, it is removed from the query. For instance, if I select `Australia`, the new query is `(Animal or Plant) and not Portrait` (282 photos from many locations). When an orange property is selected, it replaces more specific properties in the query. For instance, if I select `Pacific`, the new query is `(Animal or Plant) and not Portrait and Pacific`. In this latter case, the property `Pacific` becomes red because it is in the new query, and the property `Australia` becomes orange because this generalization gives no additional photo.

2.5 Navigating Sideward

We show in this section that downward and upward navigation can be combined in 2 forms of *sideward navigation*. From the previous query `Australia and (Animal or Plant) and not Portrait`, we first select the property `Plant` to reach the query `Australia and not Portrait and Plant` (6 photos). This is our starting point for sideward navigation.

At this point, I see that 1 photo has also the type `Landscape`, which interests me. So I select this property (downward navigation), and as the result has only 1 photo, I generalize it by removing the property `Plant` from the query (upward navigation). We have done a sidestep from Australian plants (6 photos) to Australian landscapes (80 photos), replacing in the query the property `Plant` by the property `Landscape`. From there, I perform a new sidestep from the property `Landscape` to the property `Building`, now watching 28 photos of Australian buildings. These steps are suggested and supported by photos sharing two properties. This illustrates the relevance of assigning several types to photos, which is very common in this photo context. The same would apply to persons visible on photos, as a photo often shows several persons.

The same does not apply to locations, as a photo cannot be taken in 2 incomparable locations (e.g., in Australia and Europe). However it is still possible to navigate sideward among locations, through the taxonomy of locations. Suppose I want to find building photos from other locations. I first generalize `Australia` by `Location` in the query (upward navigation), and then browse suggested locations before selecting `Spain` (downward navigation). I thus have done a sidestep from Australia to Spain, and find 18 photos, which appear to be mainly churches taken in the north-west of Spain in 2003.

The latter form of sideward navigation is a downward-upward combination, and can be qualified as *contextual* because it relies on objects sharing some properties. The former form of sideward navigation is an upward-downward combination, and can be qualified as *logical* because it relies on subsumption relations between properties.

2.6 Querying by Formulas

Most useful queries can be reached by a succession of navigation steps, but not all. Indeed the logic allows the expression of string patterns (e.g., on events) and arbitrary intervals (e.g., on dates), and the navigation trees cannot display them all. However it is always possible to use these patterns and intervals by directly editing the query. For instance, suppose I want to find ICFCA-related photos, I enter `event contains "ICFCA"` in the query field, and find myself in the same situation as if I had selected the corresponding property in the navigation trees. I find that the 68 photos in the extent are scattered in 3 different years (2004, 2006, 2007) and in 3 different locations (Dresden, Clermont-Ferrand, Sydney), and they show people from the FCA community. I can further refine my search to photos taken since 2006 by modifying the query into `event contains "ICFCA" and date in 2006...` I now find that both year 2004, and location Sydney have disappeared from the navigation trees as they are no more relevant to the new query. The result can be seen in Figure 2.

It can be observed in Figure 2 that the 2 properties used in the query are now visible. Each time a new property is given by a user, it is inserted in the property trees for future use. Indeed, if a user has found this property useful, she or another user may well find it useful on another occasion. Now, if a user is not satisfied by this behaviour, she can still hide it, as well as any other property

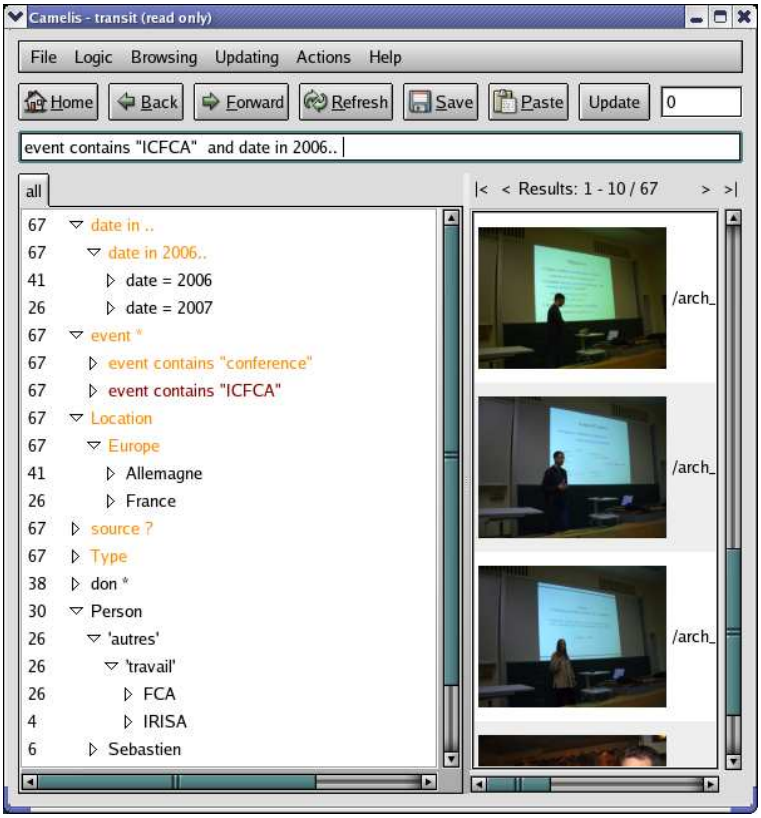


Figure2. Another screenshot of the graphical interface of CAMELIS.

by the way. Conversely it is possible to directly insert a new property, without having to use it in a query. This possibility for users to show and hide properties at will help them customizing navigation trees to their taste.

2.7 Querying by Examples

In all previous sections the query is modified either by selection of features, or by direct edition. In this section we present how a query can be determined by the selection of a subset of photos, thus supporting querying by examples. The principle is to make the query be the intent of the subset of photos.

Definition 3 (intent). *Let $K = (\mathcal{O}, \mathcal{L}, d)$ be a context, and $O \subseteq \mathcal{O}$ be a subset of objects. The* intent *of O is the set of most specific properties that are shared by all objects in O:*

$$intent(O) = Min_{\sqsubseteq}\{p \in X \mid \forall o \in O : \exists d \in D(o) : d \sqsubseteq p\},$$

where X is the set of visible properties in the property trees.

Suppose I start with the query `Australia and not Portrait`. While browsing photos in the result, I see interesting photos of buildings (e.g. 2 photos of the Opera, and 1 photo of the Harbour Bridge), and I would like to find more. By selecting them I move to a new query that is the conjunction of the properties of the intent of those 3 photos. As usual with this form of navigation, the intent query is very specific so that I get no more photos. The properties making the intent query are red in the trees, and I find some of them are very specific: e.g., a precise date (22 feb 2004), a precise location (Sydney). At this stage, I can use upward navigation to generalize the query. By removing in the query properties related to date and event, the query becomes `Sydney and Building`, and I find 29 photos. I further generalize the query by replacing the property `Sydney` by `Australia`, but find no more photos. One more generalization by removing the location, and I now find 933 photos of buildings around the world, mostly in Europe. I can also navigate downward to find photos of modern buildings, or sideward to find buildings in different countries.

A special case of querying by examples is when selecting only one photo. Then there is only one object in the extent, because there is enough properties to uniquely characterize each photo, and the query contains all the object properties, which are more easily read as red properties in the navigation trees. So this is an easy way to access the full description of any object.

3 Describing Photos

This section shows how the context that is used in Section 2 can be built in a reasonably efficient way. An important practical need is that this building can be done incrementally upon the arrival of new photos, and that everything that is done can be undone. The parts of a logical context that can be updated are (1) the set of objects (i.e., adding and removing photos), (2) the description of objects (i.e., adding and removing properties to objects), and (3) the taxonomic parts of the logic (i.e., moving a property downward and upward in a taxonomy).

3.1 Importing Photos along with Intrinsic Properties

Suppose after I took part to CLA'07, I get a new folder of photos taken during this event. In order to add these new photos to my photo context, I apply the command `Import files` to this folder so that each photo it contains becomes a new object in the context. These new objects comes with an initial description that is automatically computed from the file location and contents. The properties making this initial description are called *intrinsic*. The intrinsic properties of photos are the file location as a string, and the date. From there it is easy to select the newly imported photos by setting the query to the appropriate file location property (e.g., `file contains "My Photos/CLA2007/"`).

3.2 Adding Extrinsic Properties to Photos

The state of the art in image analysis [DLW05] makes it possible to make intrinsic low-level properties of photos, such as orientation, intensity, dominant colors or textures [ML02]. However, most high-level properties such as event or visible persons, which are the most useful, cannot be determined automatically from their contents, and have to be given manually by users [HSS02]. Those properties are called *extrinsic*. In fact there is no strict borderline between intrinsic and extrinsic properties. For instance, the location could be made instrinsic with the help of a GPS-equiped camera and a geographical information system, but these features are rarely available. Some properties (e.g., sunset) could be made instrinsic, but probably not all. The borderline is fixed as a trade-off between the cost of manually giving properties, and the cost of developping property extraction algorithms. The advantage of extrinsic properties is that they can be customized at will to the needs of users, and the design of the interface make it efficient enough as I experienced with the rich description of more than 5,000 photos.

The principle for efficiently giving new properties to photos is based on copy and paste. A set of photo thumbnails is first selected, and is then pasted on a set of properties, which can be either selected in the navigation trees, or directly entered in a text field. Removing properties is simply done by pasting on the negation of these properties. All my new photos of CLA'07 have the same event and location, so I paste all of them into `event is "conference CLA'07" and Montpellier`. Both properties are new, and are then inserted in the property trees: `event is "conference CLA'07"` is placed under `event contains "conference"`, but `Montpellier` is placed at the highest level because it is a new location. The taxonomy of locations is updated with Montpellier by the drag-and-drop of `Montpellier` under `France`, which enforces a subsumption relation between the 2 locations.

Other properties, e.g. types, persons, objects, are added in the same way. When a property already exists, it is enough to find and select it in the property trees. Otherwise it is enough to write it. In the latter case, either it is a valued attribute and it is automatically inserted, or it is a taxonomic term and it can be moved once and for all in a taxonomy. I have observed that after some number of photos I less and less often need to write new properties, and that I can rely on the property trees to maintain consistency in the use of properties. Of course the fact that I am the single user helps a lot to have a consistent vocabulary, but we could imagine a collaborative system under the principles of WIKIPEDIA or FLICKR to incrementally develop shared taxonomies.

4 Related Tools

Among applications for organizing and retrieving photos, FLICKR and PICASA are among the most famous. In PICASA, the organization of photos is limited to making albums as collections of photos. So each album can be seen as a property, and because a photo can be put in several albums, a photo can be given several

properties. However there is no hierarchy between those properties, and they cannot be combined in queries: e.g., it is not possible to get the intersection, union or difference between 2 albums. Navigation is limited to the usual file system hierarchy. In FLICKR, any set of properties, called tags or sets, can be given to photos, and each photo can also have a date, a geolocation, and an owner. The basic queries are conjunctions of tags, but advanced search allows for negation of tags, and an interval of dates. However, the different properties cannot all be mixed in a same query: e.g., "all photos from this author taken in this area about some subject". The navigation is limited to a cloud of tags, where the size of each tag is related to its global frequency. This would correspond to our initial navigation trees, where each tree would be limited to its root, i.e., a flat list of tags.

There also exists tools in the FCA community. BIBSONOMY [HsSS06] is similar in its working and appearance to FLICKR, except it applies to bookmarks and bibliographical references instead of photos. It formalizes the metadata as a triadic context that links together resources, tags, and users. IMAGESLEUTH [DVE06] is certainly the tool the most similar to CAMELIS w.r.t. presentation and navigation. It displays the current semi-concept, the extent as a set of thumbnails and the intent; it provides downward and upward navigation, querying by attributes, and querying by examples. It also uses *perspectives* (sets of attributes), which are in fact simple cases of 2-levels taxonomies: the 1st level is made of perspectives like `Location` or `Person` for photos, and the 2nd level is made of attributes such as concrete locations or persons. It also provides a way to reach similar concepts according to some distance. According to the definition of this distance, we can say that our sideward navigation are a way to reach such similar concepts.

The main advantages we have compared to these tools are brought by the use of logic. Logic enables to express different kinds of properties (e.g., dates, string patterns), and to organize them according to a well-defined subsumption relation. Logic enables users to create and customize several taxonomies. Logic enables to express complex queries where all kinds of properties can be freely combined with all boolean operators. This expressiveness is nevertheless accessible through navigation as illustrated in Section 2. Another major advantage of CAMELIS is to provide very informative navigation trees from any query, and to support all forms of navigation and querying : navigation downward, upward, sideward, backward and forward, querying by formulas and by examples. Finally, we successfully manage a collection of photos 10 times larger than the example given for IMAGESLEUTH, and we think this is because we do not need to compute the concept lattice. The navigation trees supporting navigation and reflecting the concept lattice are computed on the fly.

5 Conclusion

CAMELIS has greatly benefited from several years of application on my collection of photos. This makes it a mature implementation of Logical Information

Systems (LIS), and solves the problem of organizing and retrieving photos in a rich and flexible way. CAMELIS has also deeply changed the way I take and share photos with friends. I can quickly build customized diaporamas. For instance, to present my region Brittany to a group of foreigners, I selected all buildings and landscapes of this region, except those showing relatives. I am not reluctant to take photos that are irrelevant to the current event because I know I can easily find them afterwards: e.g., the photo of an animal during a conference event. This allows me to progressively gather collections of photos on various themes: e.g., I have got photos for 51 different species of animals, 17 different music instruments, and 255 named persons.

Besides photos, CAMELIS is also applied to music files (whose tags are automatically extracted as intrinsic properties), and to sets of bibliographical references (imported from BibTEX files and DBLP search results).

References

[CFRD06] P. Cellier, S. Ferré, O. Ridoux, and M. Ducassé. An algorithm to find frequent concepts of a formal context with taxonomy. In S. Ben Yahia and E. Mephu Nguifo, editors, *Int. Conf. Concept Lattices and Their Applications*, pages 243–248. Faculté des Sciences de Tunis, 2006.

[DLW05] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: approaches and trends of the new age. In H. Zhang, J. Smith, and Q. Tian, editors, *Int. Work. Multimedia Information Retrieval*, pages 253–262. ACM, 2005.

[DVE06] J. Ducrou, B. Vormbrock, and P. W. Eklund. FCA-based browsing and searching of a collection of images. In H. Sch rfe, P. Hitzler, and P. Øhrstrøm, editors, *Int. Conf. Conceptual Structures*, LNCS 4068, pages 203–214. Springer, 2006.

[FR00] S. Ferré and O. Ridoux. A logical generalization of formal concept analysis. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 371–384. Springer, 2000.

[FR04] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3):383–419, 2004.

[GMA93] 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.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis — Mathematical Foundations*. Springer, 1999.

[HSS02] E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In *XLM Finland Conf.*, pages 15–27, 2002.

[HsSS06] A. Hotho, R. J schke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In A. de Moor, S. Polovina, and H. Delugach, editors, *ICCS Work. Conceptual Structures Tool Interoperability*. Aalborg University Press, 2006.

[ML02] J. Martinez and E. Loisant. Browsing image databases with Galois' lattices. pages 791–795. ACM, 2002.

[PR05] Y. Padioleau and O. Ridoux. A parts-of-file file system. In *USENIX Annual Technical Conference, General Track (Short Paper)*, 2005.