DIPLOMA THESIS

Facettice: Integrating Faceted Navigation and Concept Lattices for Visual Data Exploration

by

Benjamin Bach

born 24.02.1985 in Jena

3120853

submitted

December 8, 2010

to

Professor: Prof. Dr.-Ing. habil. Rainer Groh, Chair of Media Design

Supervisors: Dipl.-Medieninf. Dietrich Kammer, Chair of Media Design

Dipl.-Medieninf. Jan Polowinski, Chair of Software Technology

Institute of Software- and Multimedia-Technology

Faculty of Computer Science

Erklärung

Hiermit bestätige ich, Benjamin Bach, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Diese Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Leipzig, 8.12.2010

Danksagung

Ich möchte die Gelegenheit nutzen mich bei meinen Betreuern Dietrich Kammer und Jan Polowinksi sowie bei Prof. Dr. Groh herzlichst für ihre Betreuung zu bedanken.

Ich danke Felix von Zadow, Tony Gjerlufsen und Fanny Chevalier für ihre Korrekturen und Anmerkungen. Ganz besonderer Dank gilt Annekathrin Müller für ihre vielen und hilfreichen kritischen Anmerkungen und die Sorge um meine Leben ausserhalb dieser Arbeit. Großen Dank möchte ich ebenfalls meinen Eltern äußern, die mich in allen Entscheidungen und auf den Wegen hin zu dieser Arbeit mit allen Mitteln unterstützt haben.

Zum Schluss möchte ich noch einmal Fanny Chevalier, Emmanuel Pietriga, Jean-Daniel Fekete und Tony Gjerlufsen danken, die mir viel Vertrauen und Kraft für meine Arbeit gegeben haben.

Abstrakt

Computergestütztes Suchen in großen Datenmengen gehört in vielen Bereichen der heutigen Informationsgesellschaft zum Alltag. Facettennavigation, auch genannt Facettensuche, ist eine effiziente und einfache Methode um gezielt nach Objekten zu suchen und in großen Datenmengen zu browsen. Facetten sind Eigenschaften von Objekten anhand derer die Objekte gefiltert werden können. Verschiedene Techniken existieren, die den Nutzer bei der Facettennavigation unterstützen. Diese reichen jedoch oft nicht aus um größere Aufgaben damit zu erledigen. Strukturen innerhalb der Daten und zwischen den Facetten, die dem Nutzer helfen würden präzisere Suchanfragen zu stellen, werden gar nicht oder nur ansatzweise genutzt und dargestellt.

Diese Arbeit entwickelt *Facettice*, ein Interface zur visuellen Unterstützung von Facettennavigation und Datananalyse mittels interaktiven Hassediagrammen. Im Vordergrund einer Integration beider Technologien stehen folgende Punkte (1) Visualisierung von Hassediagrammen, (2) Such-History-Visualisierung, (3) Query-vorschau und (4) sukzessive Konstruktion von Hassediagrammen.

Entwickelt wurden die zwei interaktive Visualisierungen, *Facet Lattice* und *Big Smart Lattice*, die die beschriebenen Punkte umsetzen. Eine prototypische Implementierung erlaubt das browsen und analysieren von Realdaten und liefert wertvolles Feedback für weitere Entwicklungen.

Contents

List of Figures

List of Tables

1 Introduction

In modern information society, computer supported search within large data sets has become an everyday practice. The enormous amount of data and the easy accessibility do not lower the efforts in finding the right items. Yet, projects like *Linking Open Data* and *Freebase* are connecting entire data networks together creating the Semantic Web, the *web of data*.

Searching in such data sets is done by posing various queries. *Focalized search* is about retrieving objects, of which the person has a clear idea. In contrast, *exploratory search* is browsing, following links and explore the data, without such a precise intention.

Two major problems must be solved by computer interfaces in order to support exploration and focalized search, which often accompanies each other. The interface has to provide tools for formulating queries as well as for guiding the user. It also should keep track of his navigation.

A technique that supports both, is *faceted search*, also called *faceted browsing* or *faceted navigation*. It is widely used and based on *facets* and *values*. A facet represents a characteristic of objects, for example `color` which can have the possible values `red`, `blue` and `yellow`. Another facet would be `price` which has a continuous range of values. Every object in the data base can be classified under multiple facets and values. By specifying the facet values that a wanted object must satisfy, a query such as `color:=red & size:=small` is created and send to the data base.

While browsing, the user changes the query several times and obtains different sets of objects. *Query preview* aims to guide the user by indicating the result of a new query before the query has been posed. Although *query preview* techniques as well as navigation support do exist, they are insufficient for search and exploration within large data sets. For a comprehensive exploration, which contains many successive queries and changing search goals, advanced techniques and interfaces are necessary. They need to support query preview, browsing history and should additionally allow the user to keep an overview over his retrieved search results. All these techniques aim to provide additional information about the data set. The more the user knows about the data set and possible inner structures, the better he is able to direct his search and exploration.

Formal Concept Analysis is a data mining technique that reflects such implicit inner structures of data in a way they can be exploited to support *faceted navigation*. *Formal Concept Analysis* classifies and groups objects according to their characteristics. Each group corresponds to one result set of a *faceted navigation*. The compound of all these groups, or *concepts*, called *concept lattice*, finally represents the whole

search and navigation space. The relations which exist among those concepts describe navigation paths the user takes while reformulating his query.

The motivation for this thesis is to develop an interactive visualization of *concept lattices*, which should guide the user in a *faceted navigation*. An integration of *concept lattices* and *facet navigation* should help the user to keep the overview over search results and posed queries as well as to assist him posing the right queries. Also, the user should be able to make compromises in case of an unsatisfying search result. Beyond supporting search and exploration, visualizations of *concept lattices* empower experts to analyze the data and to identify trends, correlations and implications. *Faceted navigation* should be employed to chose the particular data that needs to be analyzed. The possibility to analyze data also supports the normal user in exploration and search. General visualizations for concept lattices exist, but there are no comprehensive approaches in terms of information visualization.

Faceted navigation as well as *Formal Concept Analysis* are explained in Chapter 2 of this thesis, together with common search, exploration and analysis tasks.

In order to approach an interface that exploits the potential of an integration, common visualization and navigation techniques of faceted browsers as well as tools for *Formal Concept Analysis* and concept lattice visualizations are going to be analyzed in Chapter 3. Only few existing projects do approach an integration of both. The analysis of related work should inspire a general comparison of *faceted navigation* and *Formal Concept Analysis* in Chapter 4. Both technologies are compared by different criteria such as data structure, navigation and interface. The comparison is necessary, because although implicit links between both technologies have been stated, only one explicit relation exists. The comparison should reveal major integration points which are the basis for the development of an integrative interface in Chapter 6.

In order to exemplify and getting valuable feedback for the visualizations, a real-world data set from a taxonomy on visualizations is used and explained in Chapter 5. Particular use cases for a search, exploration and analysis will also be described and tested by a prototypical implementation of the concepts.

Finally, Chapter 6 of this thesis will develop *Facettice*, an interface that integrates *faceted navigation* and *concept lattices*. It is going to comprise four different views on the data whereby two of them are interactive lattice visualizations: *Facet Lattices* and *Big Smart Lattice*. *Facet Lattices* are an approach to visualize the realization of facets within the data set. The *Big Smart Lattice* is a general navigation environment. Because the complete *concept lattice* for the data set is too big to be visualized on the whole, a major emphasis must be done on reducing the shown lattice. All views will be linked together so that the user can chose the most appropriate one for the current purpose.

Chapter 7 will explain implementation details and demonstrate the use of *Facettice* by means of real-world data.

The thesis closes with a discussion on the benefits and limitations of *Facettice* and highlights major aspects for future extension.

2 Theoretical Foundations

This chapter explains faceted navigation and Formal Concept Analysis (FCA). Section 2.1 gives an overview on general search, exploration and analysis tasks. The three terms are described as different levels of the *information retrieval process*. Section 2.2 describes faceted navigation and distinguishes it from associated terms such as *faceted classification* and *dynamic taxonomies*. The section points out strengths and limitations of faceted navigation. Section 2.3 starts with the mathematical principles and formalisms, used throughout this paper. The section then introduces concept lattices as one of two possibilities to visualize a formal context that is created by FCA. Also for concept lattices, strengths and limitations are described. Furthermore, the landscape as a metaphor for information retrieval with FCA is explained.

2.1 Search, Exploration and Data Analysis

2.1.1 Information Retrieval

The term *search* origins from french *chercier*, which also means *explore*. This, in turn, comes from latin *circare* and means "going around in circles", a metaphor that is very familiar and can be understood as *encircling* something to catch it. The Encarta Dictionary defines *search* as *"to examine something thoroughly [and (B.B.)] discover something by examination"*[1]. A definition in the Cambridge Dictionary of American English is *"to look somewhere carefully in order to find something"*[2].

The previous definitions reveal that it is hard to find a universal and exact description of what is understood as *search*. In terms of computers and digital interfaces, fine grained processes and interaction steps need to be defined. The *stratified model of information retrieval interaction* [Saracevic, 1997] describes the process of information retrieval (IR) by dividing it into three kinds of activities, which can be performed in parallel. Figure 2.1 shows these activities, *Lookup, Learn* and *Investigate*, which are increasing in their complexity. *Lookup* is best described as fact retrieval and the search for items of which one has a vague idea. It also describes navigation with interfaces using predefined navigation techniques, such as menus and links. Lookup tasks are basic and heavily supported by the computer, considering keyword search, providing navigation interfaces, scanning databases and presenting results.

[1] http://encarta.msn.com/encnet/features/dictionary/DictionaryResults.aspx?lextype=3-&search=search

[2] http://dictionary.cambridge.org/dictionary/british/search 1

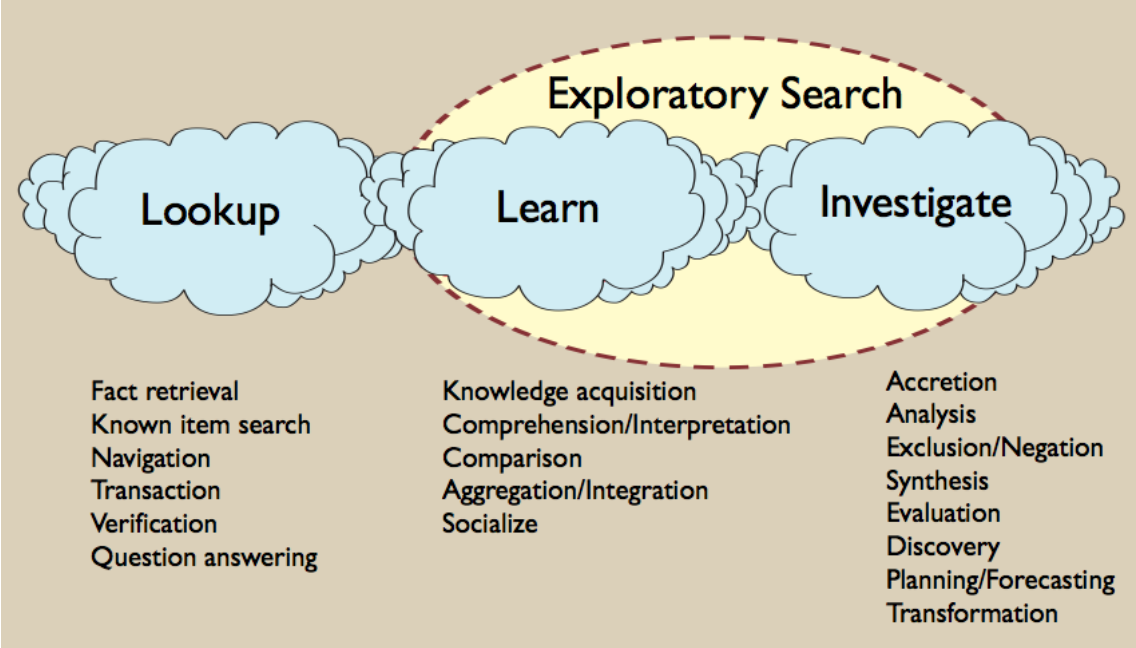


Figure 2.1: The information retrieval interaction process comprises three levels which are *lookup*, *learn* and *investigate*. The terms beneath each cloud describe particular tasks. Exploratory search is performed at the two upper layers as indicated by the yellow shape.

The next layer, *Learn*, refers to learning in general. It is a cognitive processing of information, including interpretation, comparison and aggregation. It needs repetition and ties facts together, and therefore directly involves the user into the IR process. Since it is an iterative process, information from various sources, views and of different content is aggregated and interpreted in context.

The highest layer in the information retrieval interaction model is called *Investigation*. It can be best described with analyzing and evaluating of information to draw new and own conclusions (cf. [Marchionini, 2006, p.43]).

Grounded on the division into the three activities *Lookup, Learn* and *Investigate*, Marchionini refers to *Learn* and *Investigate* as *exploratory search* as opposed to *Lookup*. Sacco opposes the term *exploratory search* to *focalized search* (cf. [Sacco and Tzitzikas, 2009, p.3]). By *exploratory search* he refers to browsing and explore relationships in a database. Basically it means the same as Marchionini with Learn and Investigate. *Focalized search* is described as quickly retrieving objects and can be compared to what Marchionini calls Lookup.

The difference between *focalized* and *exploratory search* lies in the involvement of the human as active part of the information retrieval process. While Lookup comprises basic tasks whose results depend on algorithms and user specified search terms, the results of an *exploratory search* strongly depend on the knowledge and skills of the user. *Focalized search* needs systems for navigation, query specification and result representation, whereas an exploratory search must be supported by personalization, search history, visualization and data mining methods. So far, Sacco and Marchionini use equal terms to describe a similar model of the information retrieval process. This

process is characterized by tasks of different complexity and purposes.

In addition to the previous terminology, Sacco uses a second classification of the search process. He defines *object-seeking*, *knowledge-seeking* and *wisdom-seeking* as *exploratory patterns* of an *exploratory search* (cf. [Sacco and Tzitzikas, 2009, p.3ff]). By *object-seeking*, Sacco refers to the quick retrieval of objects from databases using queries, document retrieval and text mining techniques. These object-seeking tasks also cover Marchionini's *Lookup* tasks, which indicates that a hard border between focalized search and exploratory search is hard to find.

Knowledge-seeking means to increase knowledge about a certain topic and requires a more comprehensive engagement of the user than *object retrieval*. It intends to widen the user's knowledge about a certain subject instead of retrieving just facts and objects. This pattern directly meets Marchionini's *Learn*.

Investigation tasks are called *wisdom-seeking* in Sacco's terminology. He describes the same tasks and conditions as Machionini.

The comparison of Machnionini's model to structure the information retrieval process and Sacco's model, shows many similarities between them. *Focalized search* equals *Lookup, exploratory search* comprises learn and investigate activities and *object-seeking, wisdom-seeking* and *knowledge-seeking* can be compared to *lookup, learn* and *investigate*.

These analogies suggest the profoundness of both models. Whenever referring to these models used Sacco's terminology of *object-seeking*, *knowledge-seeking* and *wisdom-seeking*, as well as *focalized search* is used.

2.1.2 The Information Seeking Process

Marchionini's and Sacco's models explain what types of search exists, how search, exploration and analysis relate and what tasks are performed in each case. Hence, it is a taxonomy rather than the description of a process. For designing a user interface for those three activities, it is important to know *what* tasks needs to be supported. But, in the same manner it is important to know *how* tasks needs to be supported, that is, in which order the user is going to do what.

An easy and widely adapted principle for describing search behavior is Shneiderman's *"Overview first, zoom and filter, then details-on-demand"* (cf. [Shneiderman, 1996]). The principle, originally referring to information visualization interfaces, also implies a search and exploration process. In information visualization, Shneiderman's principle aims at designing graphical solutions for showing an overview of the data, navigating into them and explore, and obtain details on particular objects.

In contrast to Marchionini and Sacco whose model structures information seeking into almost parallel tasks, Shneiderman's emphasizes a sequential process. Although

the visual sense is parallel in processing information, graphics can easily become overloaded. Hence, it is to be distinguished between important and less important information. The same holds for a search process and a graphical overview must serve for orientation within the data, rather than actual results.

The user subsequently narrows down his potential result set and examines remaining objects in detail. In addition to that, Shneiderman defines seven tasks for information visualization, which are *overview, filter, zoom, detail-on-demand, relate, history* and *extract*. Those tasks are an abstraction of Marchionini's and Sacco's model by referring to similar tasks. But, they can be read in a processual order; the already mentioned overview, zoom and filter, then details-on-demand is followed by a relation of known facts. A history tracks all visited items and hence the search and exploration process. The last step Shneiderman defines, *extract*, describes the ability of visualization interfaces to save portions of data so that the user can access it later. In an information retrieval process, extracting refers to the essential information.

Shneiderman' process is very simple but imprecise in to describe a search and exploration process. He therefore defines another process for information seeking, which comprises four steps (cf. [Shneiderman et al., 1997]). The steps are:

1. Query Formulation,
2. Action (running the query),
3. Review of Results, and
4. Refinement.

Here, he only describes a search, rather than an exploration. It is a far simple input-output and iteration process. Only the reviewing of results includes higher cognitive work. However, a similar but more detailed model is defined by Marchionini and White and comprises the following steps (cf. [Marchionini and White, 2008]).

1. Recognizing a need for information,
2. Accepting the challenge to take action to fulfill the need,
3. Formulating the problem,
4. Expressing the information need in a search system,
5. Examination of the results,
6. Reformulation of the problem and its expression, and
7. Use of the results.

The same processes is shown in Figure 2.2 while emphasizing the step of iteration.

Iteration has been identified as essential part of all models for an information seeking process. While iterating and posing new queries to the system, the user's intention can change and new search goals can be of interest. An other point, not addressed explicitly by Machionini, White and Shneiderman, is the consideration of all retrieved result sets together. The *berry-picking model* considers all results of each search as

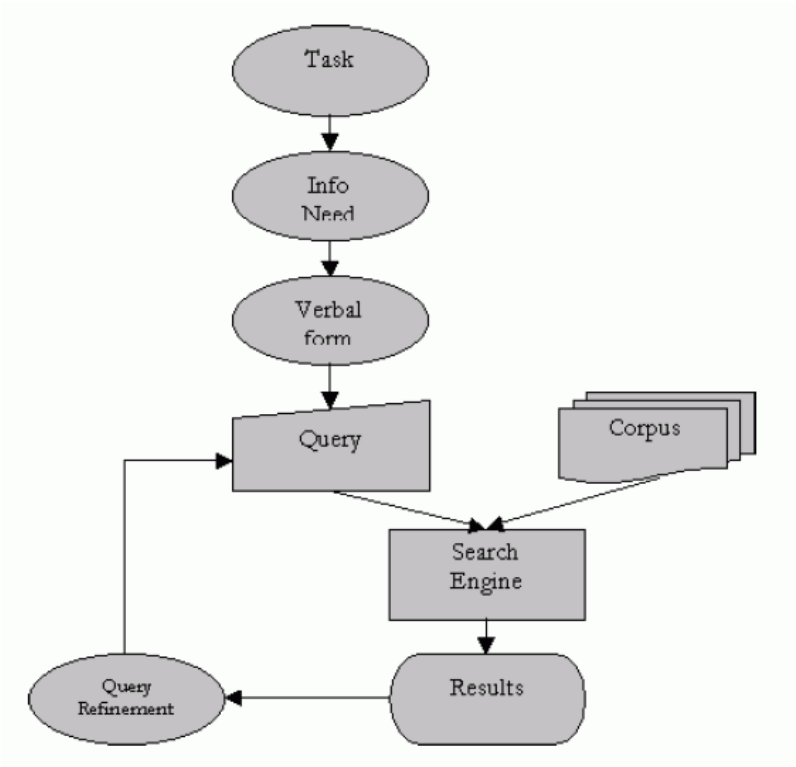


Figure 2.2: A standard model of the information seeking process. The model is reflects the ideas of Machionini and White, and Shneiderman.

the overall-result of a search (cf. [Bates, 1993]). Figure 2.3 illustrates the iteration of posing a query and retrieve documents. Thus, every query is just a step in a larger information seeking process. By that, the berry-picking model reminds of a browsing process.

Definitions of browsing, however, are as divergent as for search. Machionini describes browsing as investigative task. He argues that browsing is about knowing where and how to search. This includes to gather an overview of possibilities and order results according to their relevance for the search. A third author refers to the schema in Figure 2.1 and uses the terms *way finding* and *wander* in order to describe browsing and exploration [Morville, 2010]. Hearst describes browsing as follow links, switch between views, scan and select (cf. [Hearst, 2009]). In essence, browsing can be described as iterative and exploratory, to find interesting information and foster serendipity.

2.1.3 Visual Data Analysis

The last chapter described models and processes for searching browsing and exploration, mainly located at the two lower levels of the information retrieval model in Figure 2.1. The third and highest level is investigation and includes tasks like analysis, synthesis, evaluation and discovery, which are mainly supported by interactive

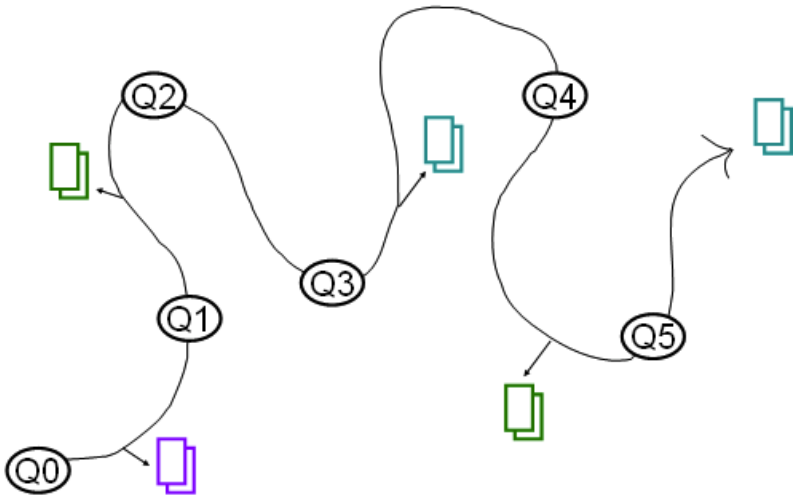


Figure 2.3: The *berry-pinking* model describes information seeking as an iterative process. $Q0 - Q5$ are queries which yield different results (represented by documents-symbols).

visualizations. This section gives a brief overview of visual data analysis.

In comparison to information visualization, visual analytics focuses on data mining methods, which analyze data before it is shown. That is, visualizations in visual analytics rely more on meta data about the data, rather than the data itself. It helps to discover trends, clusters and outliers in data sets, implicit relations and dependencies. Besides the application of extended data mining and analysis tasks before visualizing the data, visual analytics emphasizes the process of analysis on the user side. It defines the whole data analysis process as the effective sharing of labour between machine and human. While computers are used for mathematical analysis, humans are capable of evaluating and making decisions. Another term *Exploratory Data Analysis (EDA)* refers to the same process and is described in TUCKEY 1977 and ANDRIENKO AND ANDRIENKO 2006. Exploratory Data Analysis is described as a four-step process comprising the following steps (cf. [Andrienko and Andrienko, 2006, p.]):

1. formulate questions,
2. find and chose methods,
3. detect patterns/relations,
4. evaluate and decide, and
5. iterate.

Questions are related to a certain problem such as *"is there a correlation between variable A and B? And if, how big is it?"*. Then a particular method needs to be chosen that creates a picture of correlation within data. After detecting patterns and relations, they must be evaluated and investigated.

A general overview of the visual analysis process is shown in Figure 2.4. Data first is

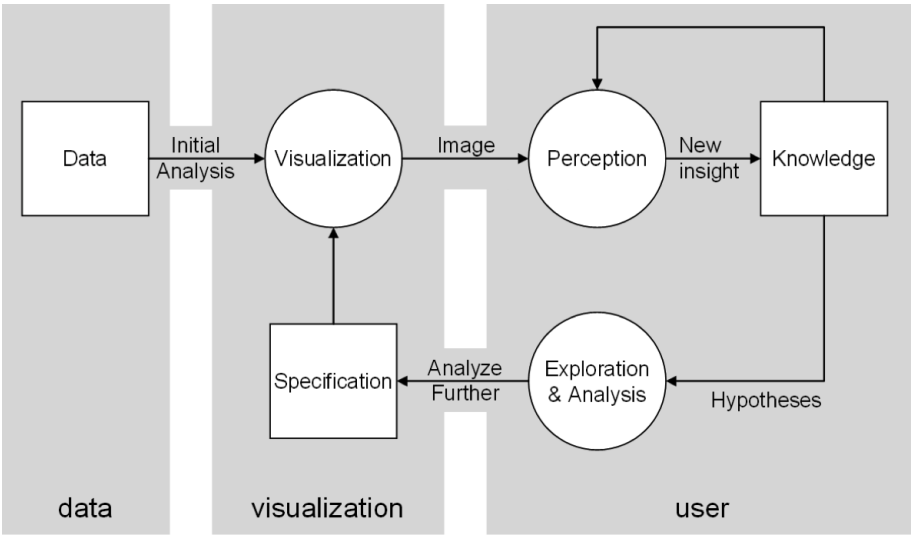


Figure 2.4: Schema of the visual analysis process. A major focus of visual analytics is on analyzing the data before it is presented. By iteration, the user changes analyzing methods and compares the results,

analyzed and mined by algorithms and then visualized. This information is not just pure data from the database but is enhanced by meta data, which results from an analytical processing. Typical analyzing and data mining algorithms are cluster and pattern analysis, association rule mining, correlation analysis estimation of trends,and supervised learning (cf. [Keim et al., 2008]). The goal is to enable the user to gain new knowledge which he can use to create questions and hypothesis, which leads to subsequent iteration in analysis and exploration. This last point is especially emphasized by Tukey (cf. [Tukey, 1977]) and underlines the processual character of analysis.

Because an analysis is not possible without exact knowledge of the data, visual analysis often involves experts from other domains into the design of algorithms and interfaces. That way, it is highly interdisciplinary. But, also among computer scientists for developing appropriate solutions for data management, data mining, visualization, human computer interaction and evaluation methods.

JONES ET AL. introduced the term Human Information Interaction (HII). While human computer interaction depends on the technical interaction media such as mouse or multi-touch displays, HII emphasizes device independence (cf. [Jones et al., 2006]). This is trivial when it comes to working with distributed information in networks where devices do variate but the access to information happens by the similar software interfaces. In the context of this work the main point about HII is about perceiving information as intractable in the same way it is visualizable.

2.2 Faceted Navigation

Faceted navigation is a navigation principle that aims for quickly retrieving a set of objects which meet a certain set of characteristics. It works for multidimensional data and is used for focalized search as well as exploratory search. The terms *faceted search* and *faceted browsing* are often used as synonyms for describing this process (cf. [Sacco and Tzitzikas, 2009] [Polowinski, 2009]). Common applications domains for faceted browsing are e-commerce application, library systems, multimedia databases and e-gouvernment (cf. [Sacco and Tzitzikas, 2009, p.263ff]).

This section explains what faceted data is, how faceted navigation work and what techniques are applied to support the user.

2.2.1 Faceted Data

The term *facet* in the context of information management was first used by Ranganathan to classify books in a library (cf. [Ranganathan, 1962]). He tried to solve the problem that many books can be classified under different terms and by different schemes. The facets he used were *personality, matter, energy, space* and *time*. He classified each book in these five categories. His intention was to support the user in finding books by providing different classification schemes as well as keeping the categorization system flexible for extension.

In computer science the term *multidimensional data* is often used to refer to data that can be classified by multiple taxonomies (cf. [Andrienko and Andrienko, 2006]). In faceted navigation, a dimension is called *facet* and divides into several *facet values*. Since a facet is considered a classification, facet values are also called *concepts*. Data objects are classified under concepts from different facets assigned. Although, it is sometimes claimed that every item can be classified under just one single concept per facet, this restriction does not hold for real world data (cf. [Priss, 2000a] [Priss, 2008]). A typical example for faceted data are cars. Cars differ in color, weight, speed and fuel consumption.

A faceted classification helps transforming heterogenous data into semi-structured. Thereby, a faceted classification just classifies objects but neither relates them directly nor states further logical restrictions on the facets, as it can be done with ontologies(cf. [Gruber, 2009]).

Figure 2.5 demonstrates the general character of multidimensional data by two common visualizations; parallel coordinates and scatterplots. In parallel coordinates, each vertical line represents a dimension which is divided according to its values. A colored line from the left to the right represents an object in the data base. The problem with scatterplots is that they can compare only two to four dimensions at once. A scatterplot matrix shows all scatterplots that exist by combining each facet with each.

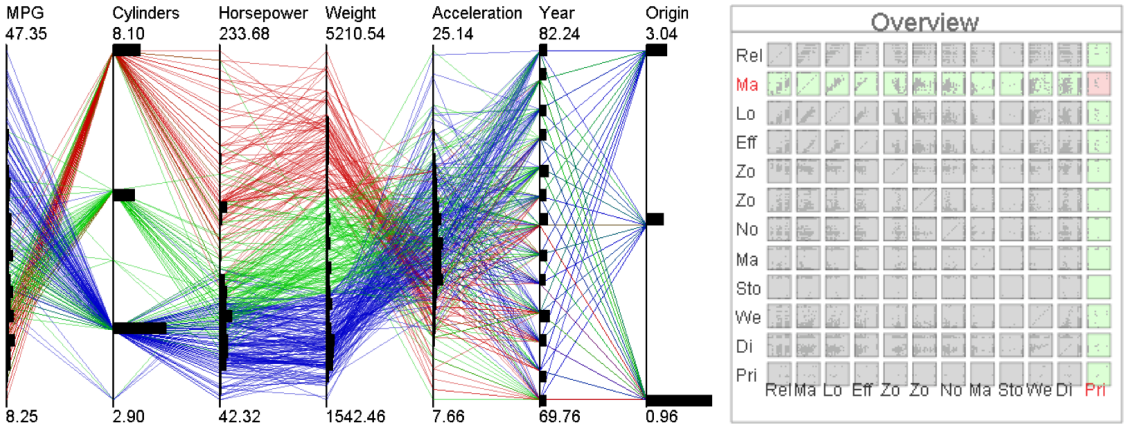


Figure 2.5: Two possibilities of visualizing multi dimensional data. The left screenshot shows parallel coordinates [McDonnell and Mueller, 2008] and the right one shows a scatterplot matrix [Elmqvist et al., 2008]. The matrix consists of scatterplots comparing each dimension (facet) to each.

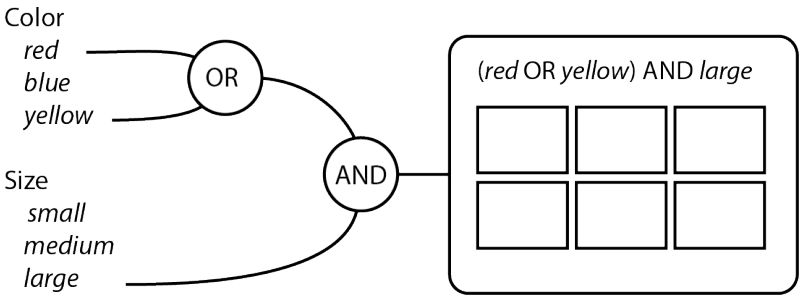


Figure 2.6: Demonstration of the query behavior of facets.

2.2.2 Navigation and Querying

Navigation within faceted data is done by restricting facets on their values. Selected facet values represent a boolean query. Values from different facets are combined by conjunction, values the same facet are combined by disjunction. Figure 2.6 gives the example for two facets `color` and `size`. The user specified the values `red, blue` and `large` and the retrieved objects correspond to the query `(red OR yellow) and large`. However, the user does not need to form this boolean query by hand. Moreover, he selects and deselects values from facets and the query is created by the rules in Figure 2.6. The user refines his query in an iterative manner. An important point about this refinement is that the search interface ought to provide feedback about further search options, i.e. omit facets whose selection would lead to an empty search result.

Yee et al. describe the process of a faceted search in three steps (cf. [Yee et al., 2003]):

1. *Opening:* gain an overview over facets and values,
2. *Middle Game:* iteratively narrowing down the result set by selecting facet values, and

3. *End Game:* Observe and evaluate the results.

The process is very similar to Shneiderman's and Marchionini and White's processes on search (cf. Section 2.1.2). Focalized search as well as browsing and exploration in terms of an exploratory search are best supported by the faceted navigation principle. Facets and values guide the user through the data set and the query mechanism provides an intuitive tool.

2.2.3 Dynamic Taxonomies

Facets as well as facet values can be put into a hierarchical order. Such a hierarchy usually consists of facets whose children are values as in the example in Figure 2.6. An extension is to define facets whose range is a value taxonomy, that is, the facet values form a taxonomy. This taxonomy is usually defined by the `is-a` relation. Simplified, a Dresden citizen *is a* saxon, *is a* german, *is a* european. There are three types of taxonomies (cf. [Sacco and Tzitzikas, 2009, p.47]) presented in the list:

- *Static mono-dimensional taxonomies* are simple taxonomies. There is only one taxonomy defined on the data set and every item is classified under one concept, as it is the case in biology. Objects can be classified under every concept of the taxonomy.
- *Multidimensional (static) taxonomies without concept composition* describe multiple independent taxonomies on the same set of items. An item is defined under exactly one concept in each taxonomy and each taxonomy represents one facet. It is not possible to combine concepts as demonstrated in Figure 2.6.
- *Multidimensional taxonomies with concept composition* are also known as dynamic taxonomies (cf. [Sacco, 2000, Sacco, 2007, Sacco and Tzitzikas, 2009]). A dynamic taxonomy is a taxonomy that allows an object to be classified under arbitrarily concepts. Such taxonomies allow concept composition by boolean queries, that is they calculate new concepts on demand.

While the basic idea of faceted search is multiple facets and a faceted classification, dynamic taxonomies are a user-centric approach to faceted search. A dynamic taxonomy is adapted each time the user changes the query, that is, she selects or deselects a facet value. By the adaptation, those taxonomy concepts, that do not occur within the new result set, are excluded from the resulting reduced taxonomy. Figure 2.7 shows the adaptation process. Items 2 and 3 are considered the *extent* of concept c, while the concept c, d and f are the *intent* of object 3[3]. Thus, all concepts are removed from the taxonomy that are not part of the intent of the extent of c. As seen from Figure 2.7, the user can only select further concepts that are present. This prevents her from empty result sets (for example, `b AND f`) and

[3] Sacco et al. distinguish between *shallow extension* that comprise only the direct items assigned to a concept, and *deep extension* by which all items of all child concepts are meant (cf. [Sacco and Tzitzikas, 2009, p.6].

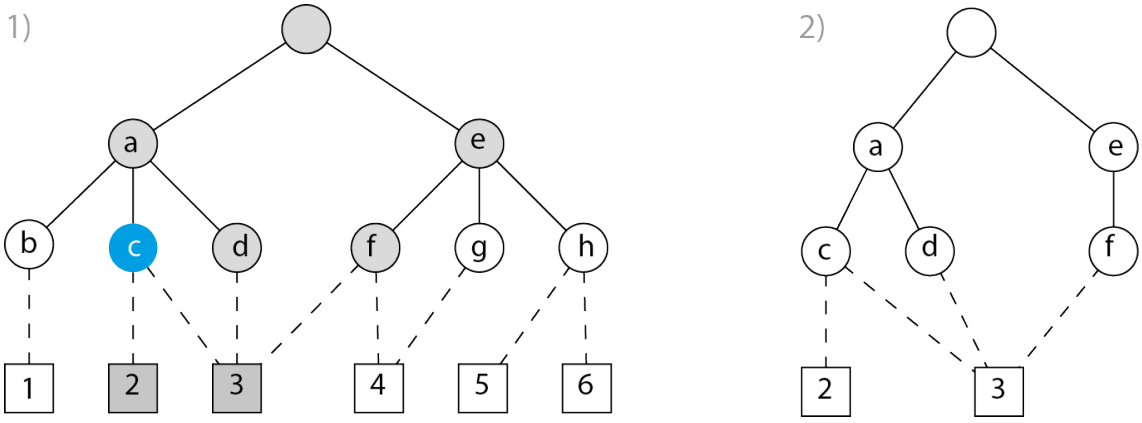


Figure 2.7: Dynamic Taxonomies are adapted when a facet value is selectd. On the left, the complete taxonomy is shown with the concept *c* that is going to be selected by the user. The right tree is the adapted taxonomy containing only those concepts that occur in the extension of *c*. The taxonomy on the left has a resolution of 1.5 items per concept.

makes further selection possibilities clearer. In the right taxonomy in the example only *d* or *f* can be further selected.

An other advantage of dynamic taxonomies in comparison to static ones, is the *maximum resolution*. This value indicates how many items on average remain under each concept. Sacco et al. report a maximum resolution of 10-20 in order to provide a user friendly taxonomy (cf. [Sacco, 2000, p.474]). Due to concept composition, dynamic taxonomies achieve a better maximum resolution, than taxonomies without concept composition.

2.2.4 Interface and Benefits

faceted browsers are applications that enable a user to perform faceted search. Popular examples are Flamenco [fla,], Longwell [lon,], Exhibit [exh,] and Camelis (cf. [Ferré, 2009]). The most common Interface for faceted browsers are lists of facet values grouped according their facets. This simple approach has been extended by many others. Some solutions provide sliders for selecting ranges, improving the visualization of facet taxonomies (cf. [Ferré, 2009]), improving the facet management (cf. [Dachselt et al., 2008]) or proving feedback about the data behind the corresponding facets (cf. [Stefaner and Müller, 2007]). Chapter 3 contains examples of advanced faceted browser interfaces.

General components of a faceted browser are (1) facet lists, (2) breadcrumbs and (3) result representation as represented in Figure 2.8. The facet lists shown on the left with their values. They serve as presentation of the taxonomy as well as for selecting the facet values. If a dynamic taxonomy is used, the lists are adapted and unused concepts are removed or faded (cf. [Polowinski, 2009]). The example also shows numbers next to each facet. They indicate how many objects remain under this concept.

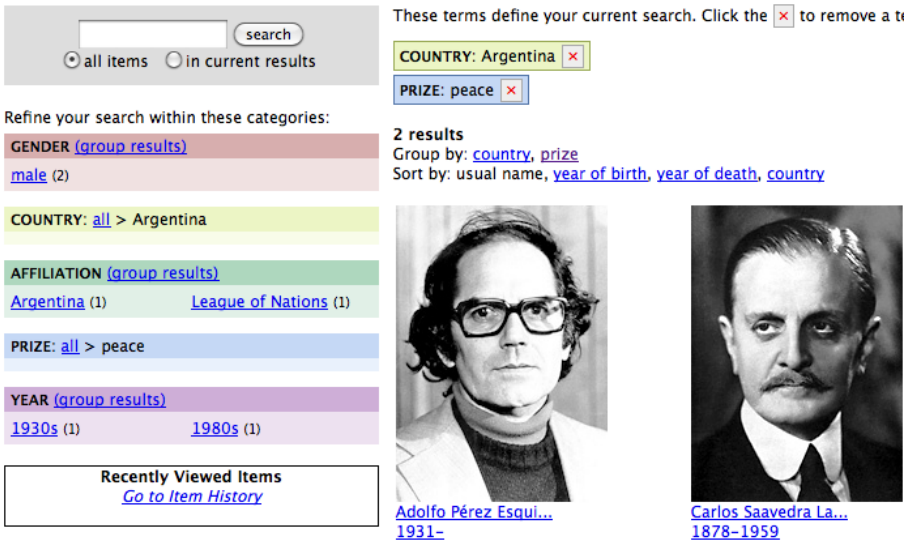


Figure 2.8: Components of a faceted browser interface at the Flamenco browser. Left are facets and values and on the right side are the results presented. Above the results is the breadcrumb section, comprising two values.

Breadcrumbs serve as search history and show the values the user has selected (cf. [Hearst, 2009, p.163ff]). In Figure 2.8 they are shown above the results and categorized by their facet. Search results are usually presented in a list and can be grouped according different facets (cf. [Polowinski, 2009]).

Challenges for future interfaces for faceted browsing are novel presentation techniques for facets and advanced mechanisms for building complex boolean queries (cf. [Polowinski, 2009]) as well as dealing with many facets and semantic web data. The latter is generic data, which strongly influences the interface design. In reference to the information seeking process, a big challenge lies in extending faceted browsing to the stage of *investigation*.

The main advantages of faceted browsing and dynamic taxonomies in order to search and browse large data sets are summarized in the following:

- Simple but powerful way for describing and classifying data
- Still works in very large data sets
- Data overview and navigation guidance due to predefined facets.
- Arbitrary amount of facets applied in parallel,
- Easy boolean query formulation by hiding the fact that it is an actual boolean query,
- Narrowing down and widening result sets by adding or removing simply one concept,
- No empty result sets,
- Low resolution because of concept composition,

- Ability to search according to different characteristics and dimensions,
- Usage of value taxonomies,
- Simple way of maintaining and extending the classification.

2.3 Formal Concept Analysis

Formal Concept Analysis (FCA) is a data analysis method from applied mathematics. It derives groups of similar items and put those groups, called *formal concepts*, into an order relation. FCA is conducted on a *formal context* that divides into *formal objects* and *formal attributes*, whereby attributes are assigned to objects. Bascically a *formal concept* is defined by a set of objects sharing the same subset of attributes. Attributes, objects and the assigned relation together form a *formal context*. Figure 2.9b presents an example context by means of an attribute matrix. The rows of the matrix represent formal objects, while the columns correspond to formal attributes. A cross in the matrix assigns a formal attribute to a formal object.

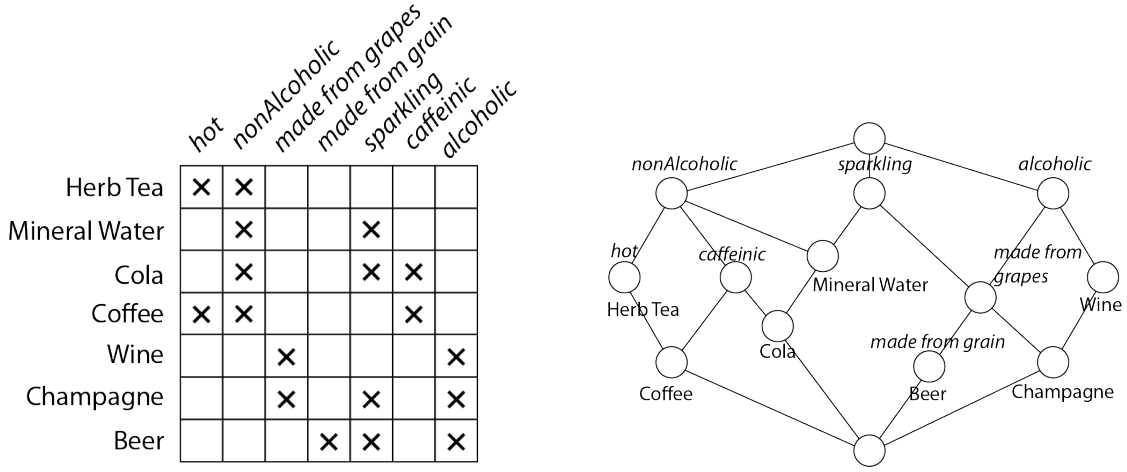
The order relation between the concepts is assymetric, transitiv and reflexive. It is called a sub-concept relation and creates a *concept lattice*. The lattice in Figure 2.9 represents all formal concepts and sub-concept-relations within the example context. It is read from top to bottom whereby the most top concept is called *supremum*, *universal concept* or *top concept* and the lowest on is called *infimum*, *contradictory concept* or *bottom concept*. The sub-concept relation between two concepts is always directed, so that the upper concepts are super concepts of lower ones.

The lattice uses a particular labeling strategy. Formal attributes are written in italic letters and are placed above the concepts. Objects are placed below. Due to the transitivity of the order relation, every object and attribute is just labeled once. the corresponding concepts are called attribute concepts and object concepts.

An attribute is valid in all child concepts of the attribute concepts, while an object is contained in every parent concept of the object concept. For example, Figure 2.9b shows that `Cola` is `nonAlcoholic`, `caffeinic` and `sparkling`. Also, every object that is `hot`, is `nonAlcoholic`. Hot drinks can be `caffeinic`, but are never `sparkling`.

Besides data mining and visual analyitics, FCA is used for knowledge representation in artificial intelligence, database and ontology engineering, class hierarchy design, software design (cf. [Stumme et al., 2002]) and linguistics.

The next section introduces a formalization and definition of terms, relationships and notions for FCA. They are mostly taken from GANTER AND WILLE 1999 to whom is also referred for proofs as well. The notion of sets and variables is taken from the same book. After that, a section on reading and advantages of concept lattices follows.



(a) Attribute Matrix (b) Concept Lattice

Figure 2.9: Two representations of a the same formal context on drinks. The attribute matrix is more an internal representation but can be used for visualization as well.

2.3.1 Mathematical Foundations

A formal context $\mathbb{K} := (G, M, \mathcal{I})$ consists of two sets G of objects and M of attributes and an *incidence relation* $\mathcal{I}$[4]. $g\mathcal{I}m$ for $g \in G$ and $m \in M$ means that "the object g *has* the attribute m".

For a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$ one defines

$$\begin{aligned} A^{'} &:= \{m \in M \mid g\mathcal{I}m \text{ for all } g \in A\} \\ B^{'} &:= \{g \in G \mid g\mathcal{I}m \text{ for all } m \in B\} \end{aligned} \tag{2.1}$$

A formal concept of the context $(G, M, \mathcal{I})$ is a pair (A, B) with $A \subseteq G$, $B \subseteq M$ and $A^{'} = B$ and $B^{'} = A$. It follows that $(A, B) = (A^{''}, A^{'}) = (B^{'}, B^{''})$. A is called the *extent* and B the *intent* of the concept (A, B). The relation between $A^{'}$ and $A^{''}$ or $B^{'}$ and $B^{''}$ is called *Galois Connection*.

If (A_0, B_0) and (A_1, B_1) are two concepts of a formal context, (A_1, B_1) is called *subconcept* of (A_0, B_0), if $A_1 \supseteq A_0$ and $B_1 \subseteq B_0$. Equaly (A_0, B_0) is called *superconcept* of (A_1, B_1) and one writes $(A_1, B_1) \leq (A_0, B_0)$. The operator $\leq$ is called hierarchical order (or simply *order*) of the concepts. The set of all concepts of $(G, M, \mathcal{I})$ ordered in this way is denoted by $\mathfrak{B}(G, M, \mathcal{I})$ and is called the *concept lattice* of the context.

An important metric is given by the *confidence*-function $conf(B_1, B_2)$. The confidence of an attribute set B_1 in reference to another one B_2 indicates how many objects from the extent of B_1 are within the extent of B_2. It follows that for $B_1 \subset B_2 \Rightarrow conf(B_2, B_1) = 1$ which can be used to state that a concept implies all of its upper neighbors. Obviously the confidence function is not symmetric which means that for $(B_1', B_1) > (B_2', B_2) \Rightarrow 0 < conf((B_1', B_1), (B_2', B_2) < 1$. The value 0 is not possible because this would deny a child concept relationship and the

[4] G and M origin from German "Gegenstände" for objects and "Merkmale" for attributes

value 1 means that $B_1' = B_2'$ which is equivalent to $B_1 = B_2$.

The following relations exist between intents and extents within the formal context. For $A, A_1, A_2 \subseteq G$ and $B, B_1, B_2 \subseteq M$:

$$\begin{aligned} &1) A_1 \subseteq A_2 \Rightarrow A_2^{'} \subseteq A_1^{'} && 4) B_1 \subseteq B_2 \Rightarrow B_2^{'} \subseteq B_1^{'} \\ &2) A \subseteq A^{''} && 5) B \subseteq B^{''} \\ &3) A^{'} = A^{'''} && 6) B^{'} = B^{'''} \\ &7) A \subseteq B^{'} \Leftrightarrow B \subseteq A^{'} \Leftrightarrow A \times B \subseteq \mathcal{I} \end{aligned} \tag{2.2}$$

These equations are essential for building a concept lattice and understand its meaning. The next section explains the construction of concept lattices in detail

2.3.2 Formalization of Concept Lattices

A concept lattice represents the order relation among all formal concepts of a formal context. From (2.2) the following two equations can be stated.

$$\begin{aligned} \left(\bigcup_{j \in J} A_j\right)^{'} &= \bigcap_{j \in J} A_j^{'} \\ \left(\bigcup_{j \in J} B_j\right)^{'} &= \bigcap_{j \in J} B_j^{'} \end{aligned} \tag{2.3}$$

These equations say that the intersection of all intents of object sub sets are the same as the intent of the union object set. Thus, by adding an object to a set of objects, it follows from 1) in (2.2) that the intent is narrowed. The same holds for extents, as shown in the second formula. By adding an attribute to a sub set of attributes, it follows from 4) that the extent contains fewer the same objects. Adding attributes to sub sets of attributes is the idea behind a concept lattice. Starting with the universal concept, concept for concept is created by all possible sub sets of attributes and their extents.

It leads to a lattice where the distribution of objects is *supremum-dense* and the distribution of attributes is *infimum-dense*. *Supremum-dense* means that all objects are part of the universal concept and are successively "removed" from the lattice while creating new concepts. This removing is the consequence of equation (2.2) 4). On the other side, an *infimum-dense* distribution of formal attributes means that the intents grow while approximating the infimum concept. This results from equation (2.2) 1).

Infimum and supremum play a special role in such a lattice. The *Basic Theorem on Concept Lattices* defines a complete concept lattice $\mathfrak{B}(G, M, \mathcal{I})$ as a lattice in which infimum and supremum are given by[5]:

[5] Taken from [Ganter and Wille, 1999, p.20]

$$\begin{aligned}\bigwedge_{t\in T}(A_t, B_t) &= \left(\bigcap_{t\in T} A_t, \left(\bigcup_{t\in T} B\right)''\right)\\ \bigvee_{t\in T}(A_t, B_t) &= \left(\left(\bigcup_{t\in T} A_t\right)'', \bigcap_{t\in T} B\right)\end{aligned} \tag{2.4}$$

The infimum of a concept lattice is the formal concept the objects of which are the intersection of all extents in $\mathfrak{B}$, that are objects that occur in every formal concept. The intent of the infimum are attributes that all these objects have in common. As indicated in the formula, the intent of the infimum is the intent the extent of the union of all concept intents. That is are all attributes that all objects have in common that have at least one attribute assigned. The infimum in Figure 2.9b is empty which means that formal objects exist which satisfiy all attributes.

The supremum of a concept lattice is defined by all attributes that are assigned to all objects in the formal context. Equation (2.4) shows that the intent of the supremum equals the union of all intents. The supremum in the lattice in Figure 2.9b has an empty intent, which indicates that no attribute is assigned to all objects.

In respect to at least two concepts one defines the *join* as the lowest upper concept of the two concept and the *meet* as the greatest lower bound. The join of the concepts labeled `hot` and `caffeinic` is the concept labeled with `alcoholic`. The meet of the two concepts is the infimum.

The amount of objects satisfying a set $B \subseteq G$ is called *support*. The value of the support-function $supp(B)$ lies between 0 and 1 and is maximal if all objects of the formal context satisfy all attributes within B. It ca hence be used to describe the amount of all objects within a concept (A, B)[6]. It follows $B_1 \subseteq B_2 \Rightarrow supp(B_2) \geq supp(B_1)$.

Another important metric is given by the *confidence*-function $conf(B_1, B_2)$. The confidence of an attribute set B_1 in reference to another one B_2 indicates how many objects from the extend of B_1 are within the extend of B_2. It follows that for $B_1 \subset B_2 \Rightarrow conf(B_2, B_1) = 1$ which can be used to state that a concept implies all its upper neighbors. Obviously the confidence function is not symmetric which means that for $(B_1', B_1) > (B_2', B_2) \Rightarrow 0 < conf((B_1', B_1), (B_2', B_2) < 1$. The value 0 does not occur because it would deny a child concept relationship and the value 1 means that $B_1' = B_2'$ which is equivalent to $B_1 = B_2$.

2.3.3 Reading Concept Lattices

Reading a concept lattice reveals much information about the formal context. The following list presents an overview of the information that can be obtained.

Sub-concept relationships The extent of a concept is the intersection of the extents

[6] Equally one can say "cardinality of the concept's extend", but it seems better to have a proper name for that.

of all parent concepts. Also the extent of a concept is the union of the extents of all its sub concepts. By this, one can consider the sub concept relation as set operations on the extents. Similarly the intent of a concept is union of all the intents of its super concepts (cf. equations (2.2) 1) and 4)).

Term taxonomy A concept lattice is a class hierarchy with multiple inheritance. Concepts on the top have a greater extent. Thus, the attributes of their intents are more general since they describe more objects. Attributes in the intent of lower concepts describe specific groups of objects. This leads to the conclusion that lower attributes are a specialization of higher ones, or sub classes. The lattice in Figure 2.9b says that caffeinic drinks are a sub class of non alcoholic drinks.

This example shows, that such a term hierarchy need to not reflect actual semantics as they are known from real world. The hierarchy is only valid for the formal context. Section 3 describes CREDO, a browser that makes explicit use of such a term hierarchy to enable browsing in large document sets.

Attribute implications. Attributes on the lower part of the lattice occur only with those above them. In mathematical terms this is equal to an implication. The lattice makes it possible to see which attributes formally imply which others. In the example in Figure 2.9b, the attribute `made from grain` implies `alcoholic` and `sparkling` (`made from grain` $\Rightarrow$ `alcoholic`$\wedge$`sparkling`). Again, this implication holds only for this particular example and to draw general conclusions, the formal context must have more objects. Finding implications among attributes is called *Attribute Exploration* (cf. [Ganter and Wille, 1999]).

Supremum and Infimum Attributes of the supremum are universally applied to all objects. Objects not part of the extent of the supremum have no attributes assigned. Objects in the infimum concept are objects having all attributes applied and their intent shows all attributes applied at all.

2.3.4 Problems with Concept Lattices

Concept lattices have two major drawbacks in terms of usability, support for novice users and the size of the formal context.

Concept lattices are generally unknown to most users without a background in FCA or mathematics. To address this problem, an evaluation of the readability of concept lattice was conducted in EKLUND ET AL. 2004 . It revealed that, although lattices are not very intuitive, users where able to understand and work with concept lattices after a short learning phase. Major problems the users mentioned were overlapping of concept labels and that empty concepts (*unrealized concepts*) where not omitted from the lattice. Concepts are unrealized if their extent is empty. Nevertheless, the majority of users stated that they would use and recommend the application if novice users would be supported more in reading and interacting with lattices.

The paper further gives a summary of visual enhancements for lattice drawing. Together with already implemented functions, feedback from the users is reported. The

following list summarizes:

- Layered background to emphasize layers within the concept lattice. The level of a node within the lattice depends on the shortest path to the universal concept. Emphasizing these level would support the user in orientation.
- Coloring nodes according their level in the term hierarchy. Same as above.
- Icons for concepts. In the test application used by EKLUND ET AL. 2004 different icons were used to distinguish between concepts with only one attribute and those whose intent is a combination of attributes. That way the users where able to distinguish between "explicit" classes of objects and "implicit" ones.
- Emphasize universal and contra-dictionary concept because they are different from "normal" concepts and mark the top and bottom of the lattice.
- Edge highlighting for guiding the user in terms of which concepts are super and sub concepts of a currently selected one. Especially when using the reduced labeling scheme as in Figure 2.9b, this enables the user to see all concepts this concept is a sub concept of.

Finally, Eklund et al. state that their evaluation was encouraging for considering concept lattices under the terms of information visualization (cf. [Eklund et al., 2004, p.64]).

The second major of drawback of concept lattices are formal contexts with many attributes, because resulting concept lattices can contain many concepts and hence many relations. Exemplification contexts usually comprise up to ten attributes, which produces lattices that can be well laid out and remain readable (cf. [Ganter and Wille, 1999]). However, real world contexts can comprise for example 8416 objects and 80 attributes, leading to possibly 2^{80} concepts (cf. [Stumme et al., 2002, p.537]). Such a concept lattice cannot be represented completely with one single lattice.

To reduce a formal context, several methods can be employed. A first class of methods is considered by mathematically reducing a context. Iceberg lattices show only the top concepts of a lattice. A concept is shown if its *support* is higher than a certain threshold. The threshold value can be adapted by the user in order to show more or less of the lattice (cf. [Stumme et al., 2002]). A support of 0% shows the whole lattice including the contra-dictionary concept.

Another mathematical method are Alpha Gallois Lattices (cf. [Pernelle and Ventos, 2003]). These lattices cluster similar concepts into a semi-concept. The value *alpha* indicates the percentage of objects that have to be equal in two concepts in order to put them together. For example, if more than 95% of all non alcoholic drinks were actually hot, those two concepts from the context in Figure 2.9b would be represented by a single concept with the intent `{hot, nonAlcoholic}`

A second class of methods reduce a context manually. One possibility is to reduce the number of attributes. Another is to define two subsets of attributes. Following that, a nested line diagram can be drawn. Such a lattice nests another one into each concept as shown in Figure 2.10. The major lattice is built upon the first set of

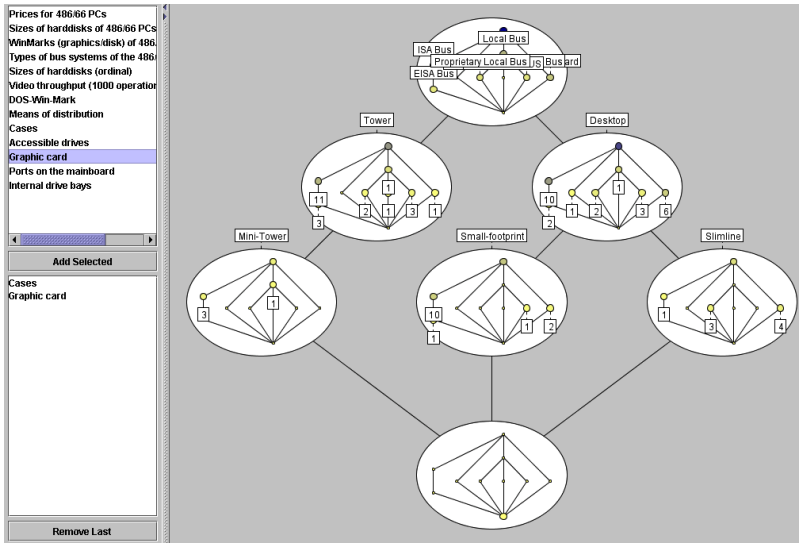


Figure 2.10: Nested concept lattices in Toscana, built from a context on computers. The major lattice is created from attributes describing case types and the inner lattices are created from attributes describing graphic cards.

attributes, while the smaller lattices are created upon the second set of attributes.

A major impact on the readabilty of a concept lattice has the layout algorithm. Basically there exist two general approaches (cf. [Eklund and Villerd, 2010]). The first is vector based and the second one is based on common graph layout algorithms. Vector based algorithms consider each attribute as a vector. A concept is positioned according to the vectors of its intent.

The main component of a graph based lattice layout is a force-directed algorithm. First, concepts are positioned according their depth in the lattice. In a second step attraction and repulsion forces are calculated among all concept relations. Such force-directed algorithms are based on physical models (cf. [Shepard,]). The edge crossing within the resulting lattice is minimized but readability can be affected due to a very horizontal position of the nodes. Figure 2.11 shows a lattice build with a force directed layout.

Conceptual Landscapes of Knowledge

Wille proposes the landscape as metaphor for knowledge processing. In terms of this landscape metaphor he defines several tasks such as exploring, searching, analyzing, investing, deciding, restructuring and memorizing (cf. [Wille, 1999]). He argues in favor of a landscape because it is directly related to the described tasks. A landscape is also traveled and observed from many different perspectives and in different scales. A landscape can be mapped by emphasizing important aspects and omitting others (cf. [Wille, 1999, p.345]). This metaphor is considered again in Chapter 6, when developing *Facettice*.

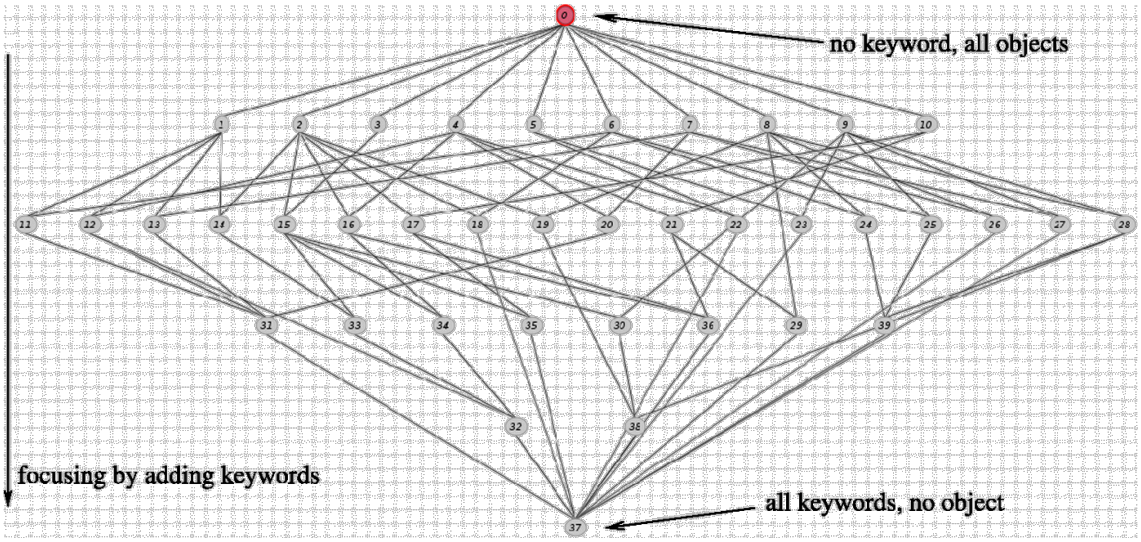


Figure 2.11: Concept lattice created from a context with 10 attributes and 329 objects, using a level wise layout approach

2.4 Conclusion

Information retrieval divides into the three levels *focalized search*, *learning* and *investigation*. Each level comprises several tasks differing in complexity and user involvement. Processes for information retrieval must be defined depending on the level, tasks and purpose of the user. Visual Data Analysis employs various techniques, mostly from data mining, in order to analyze data by means of visualization and interaction.

Faceted navigation has proven useful in many domains for quickly retrieving objects and browse large databases. The technique and corresponding interfaces are easy to use because they are intuitive and minimal. However, a major drawback is that the user sees only a part of the data base at once and browsing hints are mostly limited to indication of the amount of remaining objects behind the next facet value. By browsing the data base he may reveal relations between facet values, but in a very implicit way.

Concept Lattices are a powerful tool to observe relations between classes of objects and attributes themselves. Due to their hierarchical assembly, concept lattices serve well for an differentiates view on the data. A major drawback of concept lattice representations is the size of formal contexts. Existing solutions comprise nested line diagrams, decomposition and context reduction by restricting the considered attributes.

In the context of holistic information retrieval it raises the question of the potential of a combination of the strengths of faceted navigation and concept lattices. Using faceted navigation as navigation principle and concept lattices for visualizing the formal context. The result would be characterized by possibly covering and integrating all three levels of information retrieval. Faceted browsing can be used for focalized

search and concept lattices can provide direct visualization of the search context. Furthermore, data analysis can be conducted with the lattices in order reveal relationships. The results of such an analysis can be used for quickly retrieving objects and explore the data base.

One can differentiate between *information structure* and *navigation structure*. Information structure defines the structure of the data, that is data type, values and relations, while the navigation structure defines the ways a user can take through this information structure (cf. [Hearst, 2009, p.76] from [Morville and Rosenfeld, 2006] and [Newman and Landay, 2000]).

The landscape metaphor used by Wille to describe Conceptual Knowledge Retrieval is also worth a closer investigation.

This chapter implicitly stated some similarities and differences of faceted navigation and FCA. However, in order to investigate the potential of an integration, a more detailed comparison of faceted navigation and concept lattices must be done. The next chapter presents faceted browsers for faceted navigation and tools for browsing formal contexts or visualize lattices, or both. This aims to give an overview of trends, integration potential and the degree to which an integration of faceted navigation and concept lattices has already been attempted. The projects and their employed techniques serve as a sensible addition for a comparison of both technologies done in Chapter 4.

3 Related Work

The last chapter already mentioned major characteristics of faceted browser interfaces such as facet and value lists, breadcrumbs and presentation of the search results (cf. Section 2.2.4). Techniques to support a search and exploration process are breadcrumbs, colored facets and query preview in form of small numbers associated with the facet values. Section 3.1 presents some more advanced interfaces and techniques for faceted search.

For Formal Concept analysis, many tools can be found (cf. [pri,]). Nevertheless, most of them are concerned with algorithms and infrastructure for FCA, rather than lattice representation, information visualization or interaction. Section 3.2 classifies FCA tools into (a) software tools focusing on algorithms, (b) browsers using FCA for object retrieval and (c) visualizers that are about a visual and interactive representation of concept lattices.

Section 3.3 presents approaches that combine faceted navigation and concept lattices. The conclusion in Section 3.4 summarizes trends and techniques for faceted browsers and FCA tools.

3.1 Faceted Browsers

The faceted browsers presented in this section are analyzed in terms of facet presentation, result presentation and breadcrumb visualization. A fourth criteria is query preview, because it is a pragmatic way of supporting the user in browsing and navigating. Although query preview techniques should give a hint of the next results, it does not need to anticipate the complete state of the system, including adapted facet values and search results.

An example that partially anticipates the new system state is *RelationBrowser* [Zhang and Marchionini, 2005]. Besides simple value lists for facet presentation, it uses bar charts to show result proportions for each value. Figure 3.1 shows three facets and their values listed below. White bars indicate the total amount of objects associated with the value, while the blue bars represent the proportion of objects in the current result set. The numbers in blue correspond to the length of the blue bar. When the user hovers a facet value, all blue bars are adjusted as if the facet value would have been already selected. This method allows quick and advanced query preview without changing to much on the interface.

Search Results are presented as simple textual entries within a list, not shown in the screenshot. A major drawback of the current Relation Browser Interface is that only

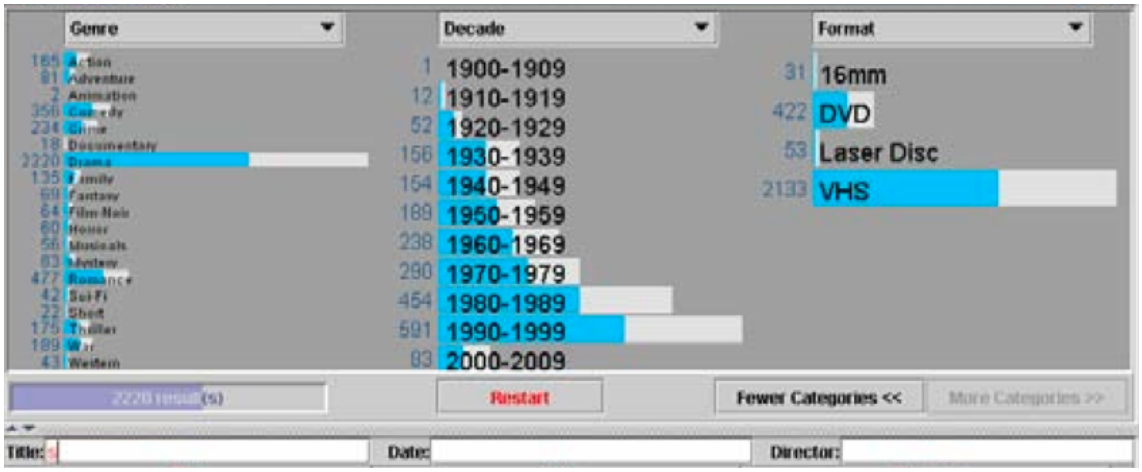


Figure 3.1: Interface of the RelationBrowser++ which uses bars associated with every facet value to indicate the number of objects remaining under the terms.

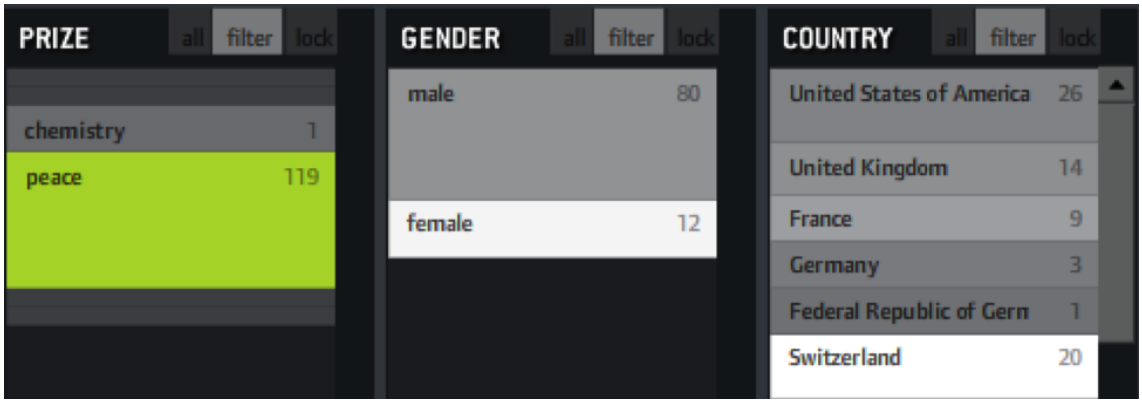


Figure 3.2: Screenshot from three Elastic Lists, each representing a facet. Facet values are represented as rectangles differing in height and brightness to depict the number of objects remaining under each facet value.

three facets can be viewed simultaneously.

Elastic Lists are similar to RelationBrowser and are integrated in a greater browsing and analysis application called *Content Landscape* [Stefaner and Müller, 2007, Stefaner, 2008]. Similar to RelationBrowser, Elastic Lists use size for query preview. Three elastic Lists are shown in Figure 3.3, each representing a facet and corresponding facet values. Facet values are represented by a rectangle differing in height and brightness. The height indicates the amount of associated items within the current result set, comparable to the blue bars in RelationBrowser. The brightness represents the proportion of these items to all items in the data base associated with this facet value. In addition to the list presentation for facets without value hierarchy, a nested view is presented if the facet contains a value hierarchy. Breadcrumbs are indicated by attributes whose rectangle is green.

If a value was selected, other lists and values are adapted immediately to the new result set. The example shows the selected value `peace` and the adapted values.

RelationBrowser and Elastic List, both indicate information about the amount of

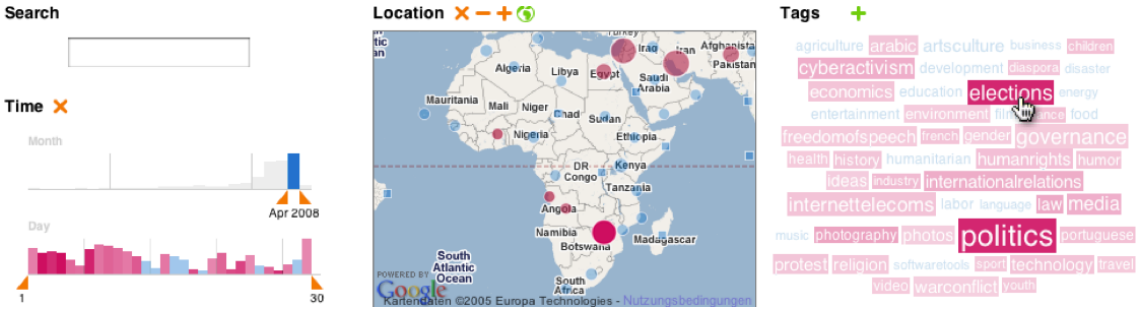


Figure 3.3: Different facet presentationd with VisGets; time line and bar chart, map and tag cloud.

objects within the current result set and their proportion to all objects. Proportions are in both cases calculated for each facet value independently. A mapping from these numbers and proportions of objects to graphical variables provides better possibilities in choosing the next navigation step.

Facets in *Visgets* are considered as diagrams depending on the semantics of facets [Dörk et al., 2008]. There is a time line combined with a bar chart for temporal facets, a map for geographical facets and a tag cloud for arbitrary ones. To support query preview, facet values differ in size according their global distribution and brightness to indicate their distribution within the current result set. Results in VisGets are presented as a list.

Those diagrams provide a more holistic perception of facets, because they express the general character, such as spatial, temporal or categorical. The diagrams indicate contextual information about the distribution and trends of values, enabling a rudimentary analysis of the data set. Because this analysis is made in a temporal or spatial context, it differs from mapping numbers and proportions to just bar charts or size.

In terms of a deeper analysis of the data set, *ResultMap* uses a tree map for representing the distribution and correlation of facet values [Clarkson and Foley, 2008]. Figure 3.4 shows an integration of this technique integrated in the *Flamenco* faceted browser [fla,]. Facet presentation, result presentation and breadcrumbs are managed by Flamenco.

A ResultMap combines two facets. Facet values of the first facet are mapped to rectangles of a certain size, corresponding to the amount of items associated with this facet value. The values of the second facet are also rectangles, but within each rectangle of the first facet. The second values are distinguished by colors as seen in Figure 3.4, while the first facet uses text for distinguishing their values. Hierarchical values can be represented by different levels of the tree map as seen in the example of `Europe`.

The advantage of the tree map for visualizing proportions and absolute values is that it best uses the screen real estate. In contrast to bar charts and geographical maps, which include a lot of white space, tree maps are compact. Another facet browser that uses a tree map is *FacetMap* [Smith et al., 2006]. It presents facets

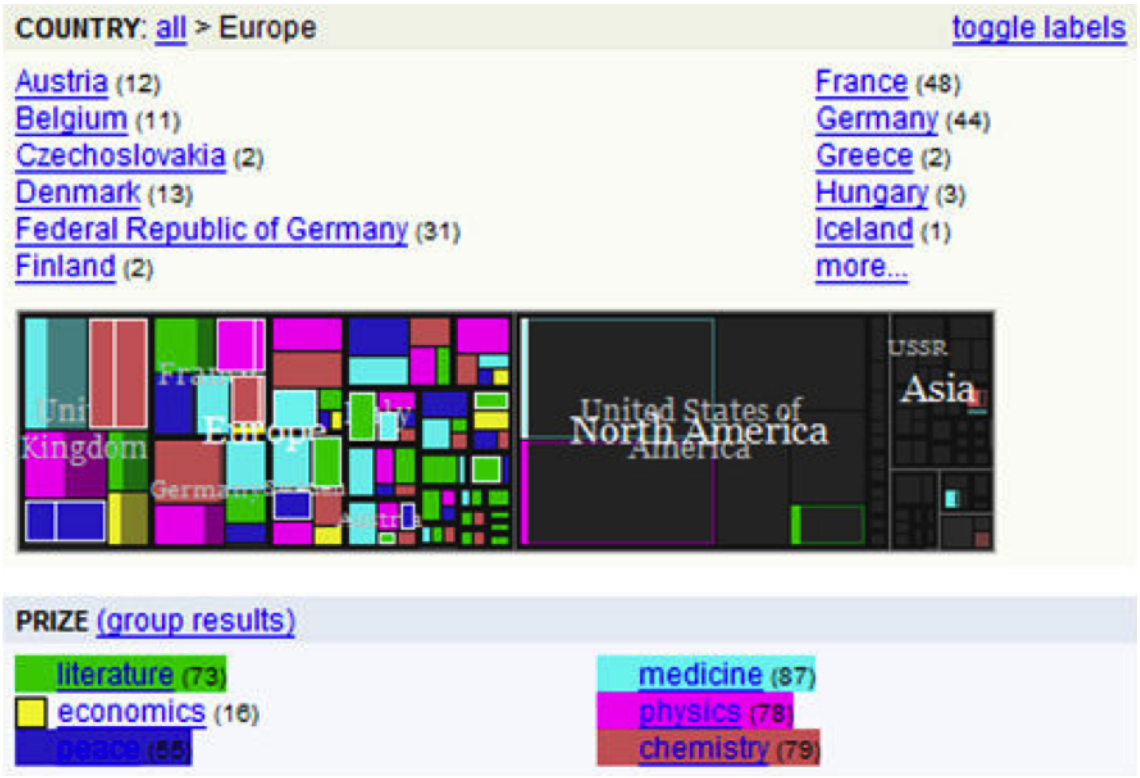


Figure 3.4: ResultMaps as integrated in the Flamenco Browser. The rectangles of the tree map correspond to the amount of objects labeled with each country while color is used to show proportion and distribution of a second facet prize.

as rectangles of a tree map, which are adapted in size each time the user changes constrains a facet.

VisGets and ResultMap are concerned with enhanced visualizations of facets, values and the distribution of results. *Faceted and visual graph navigation* is a method to directly browse within the results. It uses hierarchical clustering in order to structure the results and enable further narrowing [Tvaroek et al., 2008]. Clusters are formed by explicit structures among data such as co-authorship relations in a publication database. When searching for papers on visualization, clusters are calculated according to these internal structures. Clusters are shown as clouds and can be broken apart to see nested clusters as shown in Figure 3.5. The clouds contain facet values that occur most within the particular cluster.

The facet browser, Faceted and visual graph navigation is integrated in, consist of common interface components for facet and result presentation. The technique of combining simple and intuitive faceted browser interfaces with interactive visualizations is a promising way of searching, exploring and analyzing complex data.

Faceted browsers such as Elastic Lists, RelationBrowser++ and Result Maps use information visualization techniques to support the user in finding the right objects or exploring the data sets. In general, these techniques emphasize (a) the distribution of objects among the facet values, (b) query preview techniques, (c) presentation of and navigation within facets and (d) the direct navigation within the result set.

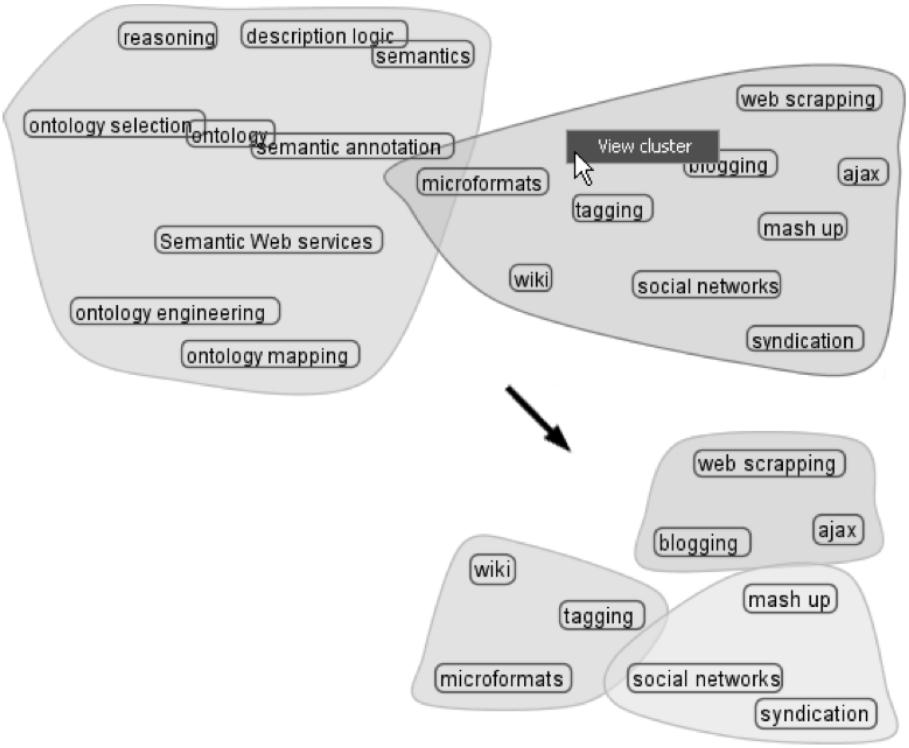


Figure 3.5: Screenshot of faceted graph navigation showing result clusters and sub clusters.

3.2 Concept Lattice Tools

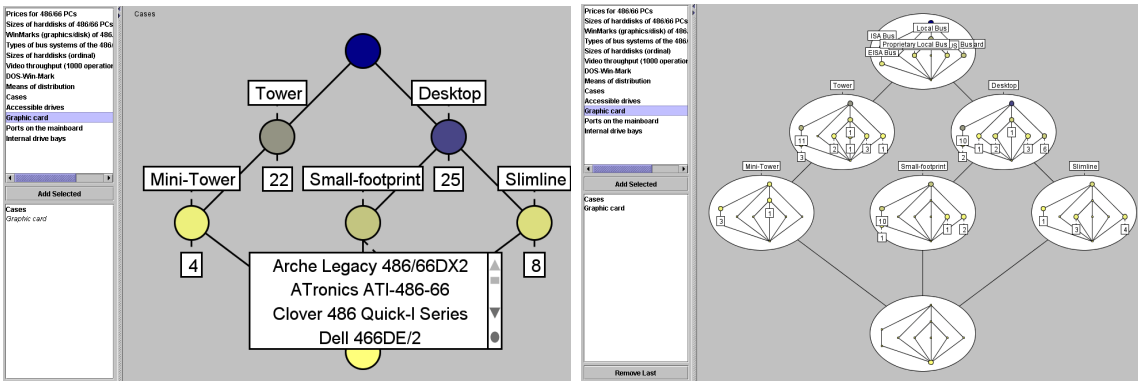
3.2.1 Software Tools

Toscana is written in Java and provides a standard visualization of concept lattices [Becker et al., 2002]. It is often integrated into other projects and follows Wille's notion of Conceptual Knowledge Landscapes. The left screenshot in Figure 3.6 shows a lattice for computer cases. The large box shows particular computer for one concept. Of particular interest are nested line diagrams as shown on the right screenshot in Figure 3.6. They are created from two different facets that can be selected from the list on the left. In each concept of the first lattice, a second lattice for the second facet is nested. Basically, this principle can be used ad infinitum. But is currently limited to two lattices in Toscana, because the tool does not provide an appropriate interface for zoom.

Other software tools are *ConExp* [Yevtushenko, 2006] and *Galicia* [Valtchev et al., 2003].

3.2.2 Browsers

Browsers based on FCA emphasize navigation though concepts. They provide navigation techniques but do not use concept lattices for representations of formal contexts. The interface uses different methods for guiding the user in a search and



(a) Usual lattice (b) Nested line diagram

Figure 3.6: Concept lattices in Toscana

exploration process that are better suited to deal with very large contexts.

This section explicitly distinguishes between the logical concept lattice and their visual concept lattice representation.

A simple but intuitive method to browse though concepts is presented with *CREDO*. It is a web based browser using Formal Concept Analysis (cf. [Carpineto and Romano, 2004]). The user starts by entering keywords to which documents are retrieved from the internet. Then, a concept lattice is generated based on important terms occurring within them.

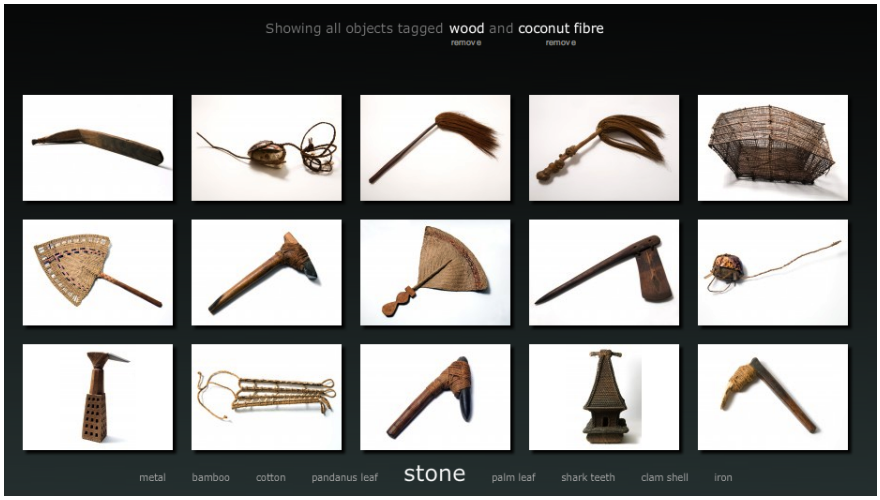
CREDO creates a horizontal tree structure by that a user can navigate through the lattice. Every path through the tree is a path through the lattice. Figure 3.7a shows the hierarchy for the initial keyword *infovis*. It shows that 14 documents labeled with `infovis` and `toolkit`, 6 are also labeled with `data`. The user can expand the tree node terms to see which other terms are associated and can furthermore select terms to filter documents. Terms in a child relationship are combined by an AND operator. A thesaurus is used for associate synonyms of keywords.

The hierarchy consists of only two levels, making it impossible to combine more than two terms (in addition to the initial keywords) and explore the full lattice. Since only 100 documents are retrieved, in order to keep calculation time short, by combining 2 terms, the remaining result set is small enough. The hierarchy is comparable to dynamic taxonomies since it hides terms that do not match in the document set. It does not, however, show attribute implications. If a document is labeled `visualization` it is not automatically labeled `toolkit`. CREDO does not create a term hierarchy comparable to a facet value taxonomy.

Another way of browsing a formal context is found is developed in *ImageSleuth* (cf. [Ducrou et al., 2006]) and adapted in the *Virtual Museum of the Pacific (VMP)* [Eklund et al., 2009]. Both projects base on the principle that each concept corresponds to an AND query of the attributes of its intent. By selecting attributes from a tag list, the user retrieves images as shown in Figure 3.7. The retrieved objects are the intent of the current concept. If the user removes or selects additional tags, the re-

- visualization (45)
- data (20)
- ieee (18)
- **toolkit (14)**
  - visualizations (6)
  - data (6)
  - interactive (5)
  - visualization (5)
  - tools (4)
  - javascript (4)
  - *other (2)*
- blog (12)
- conference (11)
- interactive (10)
- tools (9)
- visualizations (9)
- visual (9)
- papers (7)
- wiki (6)
- software (6)
- october (5)
- visweek (4)
- *other (14)*

(a) Lattice navigation in CREDO



(b) Screenshot from the Virtual Museum of the Pacific an FCA based image browser. The retrieved objects are specified by "wood" and "coconut fibre" and can be further specialized by the tags below. "Stone" occurred most within the current result set and hence does not narrow down the set significantly, if selected.

Figure 3.7

sult set changes to either a parent or a child concept. The user iteratively navigates from concept to concept. However, the user is not aware of any concept lattice or FCA mechanisms.

The navigation principle of the VMP enables to explore contexts of arbitrary size. Besides selecting attributes from a list and further result set refinement, ImageSleuth and VMP allow query-by-example which is based on the similarity of concepts. The similarity of concepts is calculated by the similarity of their intents. Since adjacent concepts within a concept lattice have already the same intent, searching for similar concepts becomes a matter of traversing the concept lattice. Figure 3.8 shows the successive traversing of neighbors of the concept S. Concepts labeled by X are direct neighbors, and concepts labeled by a stroke are neighbors of the neighbors, traversed in the next step. ImageSleuth allows the user to specify a percentage threshold for similarity.

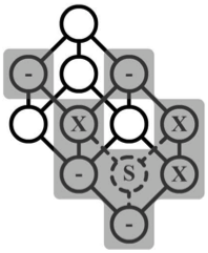


Figure 3.8: Traversing the concept neighborhood.

Browsing interfaces based on FCA do not differ much from ordinary search or exploration interfaces. The FCA process and its techniques are transparent to the user. SearchSleuth performs FCA on the results of a web search and creates a formal context from outstanding terms within the documents (cf. [Ducrou and Eklund, 2007]). Figure 3.9 shows the interface of SearchSleuth. Besides the text fields for typing keywords, it shows three categories of attributes. The attribute in red are those of the current intent and the blue ones are attribute that can be added. The attributes in gray are grouped into sets which are intents of neighbored concepts. Since there are potentially many neighbored concepts, SearchSleuth changes only one attribute at once.

Other tools like *FooCA* use attribute matrices and concept lattice representation for

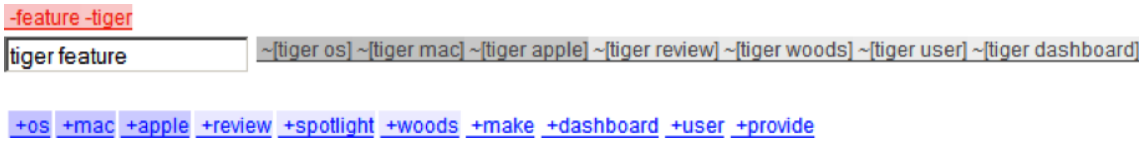


Figure 3.9: Interface of SearchSleuth showing terms for navigation to parent (red) , child (blue) or sibling concepts (gray).

showing conceptual relations between documents of a web search [Koester, 2006]. The next section presents tools that use enhanced lattice representations going beyond simple line diagrams.

3.2.3 Visualizers

The third group of FCA projects related to this paper are those concerned with a visualization of concept lattices. Toscana and ConExp also use concept lattice representations but do not provide further interaction and visualization techniques. Visualizers focus on the usability of concept lattices and consider them as primary interface for interacting with a formal context. They often integrate browsing techniques as well.

The first interactive visualizer of concept lattices was *Ulysses* [Carpineto and Romano, 1995]. It uses a layer and graph based concept lattice representation in the context of document retrieval. Figure 3.10 shows the concept lattice after the user has entered the query term `knowledge-based-systems`. In this example, this terms implies two other, shown together in the concept. All formal concepts until the universal concept are shown. The attribute names are written within the concepts with a number indicating the amount of objects.

Two techniques are of interest in Ulysses. The first is a fisheye for browsing concepts as shown in the example. The current concept is drawn in black and is bigger than others. Direct neighbors are smaller and far distant concepts are shown only by the the number representing their extent. Concepts that are even farer are omitted completely but drawn if the user changes the current focus.

The other technique Ulysses uses, is bounding. Bounding means to restrict the current search space by any direction within the concept lattice. Main directions are *parents, children* and *sibling* concepts.In the example the lattice on the right results by excluding all other child concepts of of the attribute `natural-languages`.

Mailsleuth is the first commercial FCA and Concept Lattice tool for average users [Eklund et al., 2004]. Figure 3.11 shows a concept lattice of the user's mail folders. The lattice is integrated directly into a general mail application. Green bars in the background indicate the levels of the lattice. Upper concepts represent explicit mail folders and the other ones are derived folders containing mail that from several explicit ones. Unrealized concepts are potentially possibly but do not contain objects. They are depicted by smaller nodes. Showing unrealized concepts, however, have shown to confuse users. Top and bottom concepts are differentiated by triangles to emphasize

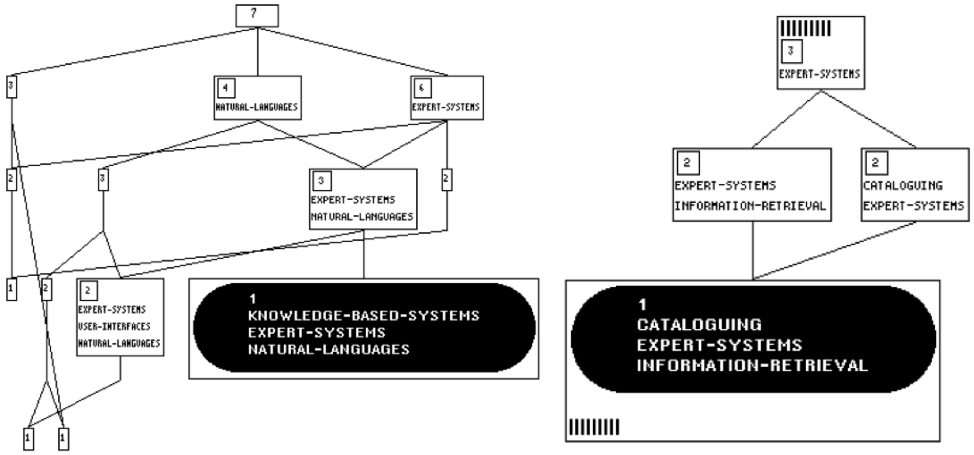


Figure 3.10: Ulysses Interface demonstrating a concept lattice with an fish-eye effect applied to the black concept. The right lattice is achieved by *bounding*, that is restricting, the search space to only parent concepts.

their special character.

A more recent approach to drive lattice visualization towards interaction and usability is made with *ConflexPlore* [MIT,]. It is written in ActionScript using the Flex framwork[1]. The core of Conflexplore is an open source framework *openFCA* deployed by the same persons [MIT,]. It implements a basic algorithm for concept calculation and draws a layered lattice refined with the Spring Graph force directed algorithm [Shepard,]. This algorithm can be update the lattice at each time. Thus, the user can move concepts and the lattice is adapted smoothly, making the representation more tangible and traceable in terms of changes.

With Conflexplore, the user can define and change own contexts by using an interactive attribute matrix, perform attribute exploration, view a list of attribute implications and finally navigate through the visual concept lattice. Figure 3.12 shows the matrix representation and the concept lattice representation. Both screens are independently and can not be seen at the same time nor used interactively by brushing and linking (cf. [inf,]).

While the matrix representation is only used to edit the formal context, the lattice representation is used for exploration and navigation. Each concept shows the name of the attribute and the object, in case there are only one in each case. Otherwise the amount is shown. In order to deal with large contexts, the user can adjust the amount of visible neighbor concepts by a range slider. Equal to Ulysses, the visible concept lattice changes when the user focusses on another concept.

The tools discussed in this section are designed for either a specific domain like MailSleuth, or are single visualizations like Toscana and Conflexplore. Also those tools have a major focus on search and exploration.

FolksonomyAnalyzer is a holistic analysis tool for folksonomies (cf. [Yang et al., 2009]). The tool is created in the context of *Conceptual Knowledge Discovery in*

[1] http://www.adobe.com/de/devnet/flex/

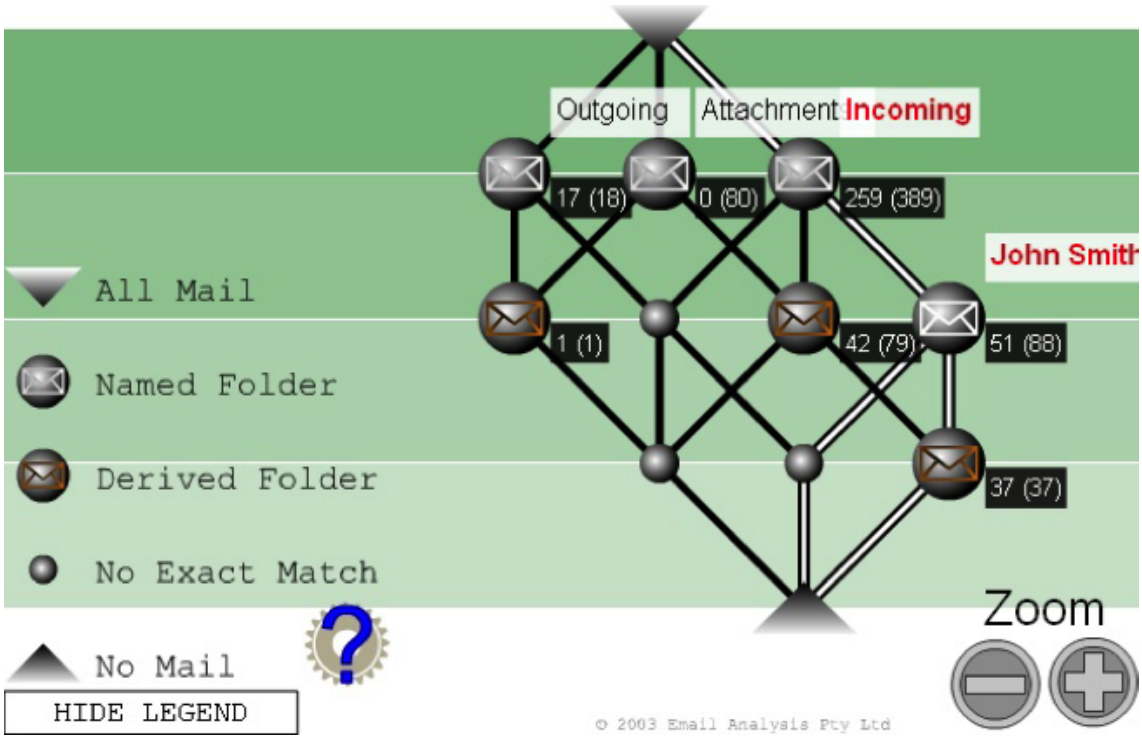
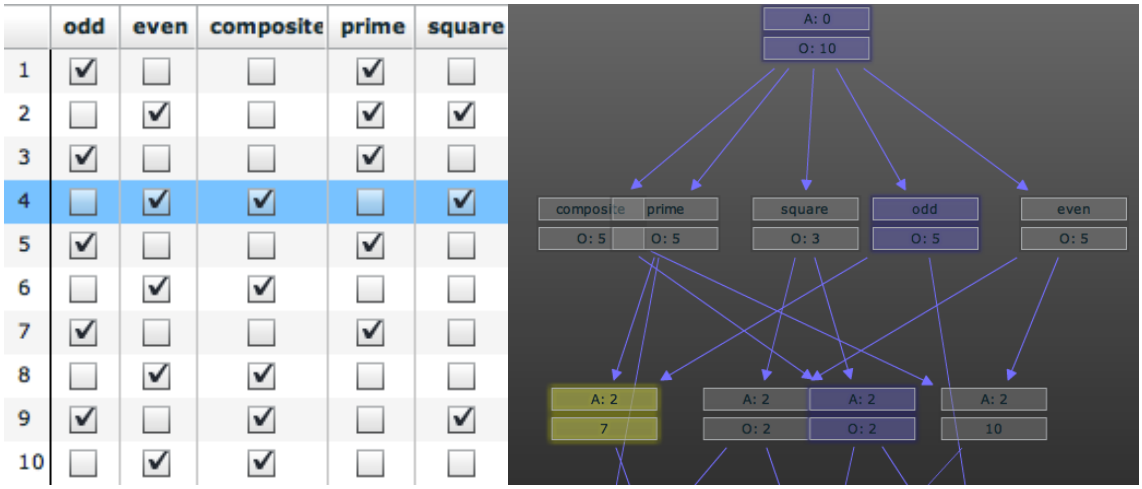


Figure 3.11: Screenshot of MailSleuth showing the concept lattice based on e-mail folders.



(a) Attribute matrix (b) Concenpt lattice

Figure 3.12: Attribute matrix and concept lattice in Conflexplore representing a context of numbers. The yellow concept in the lattice is the currently selected one and the lilac ones were previously selected.

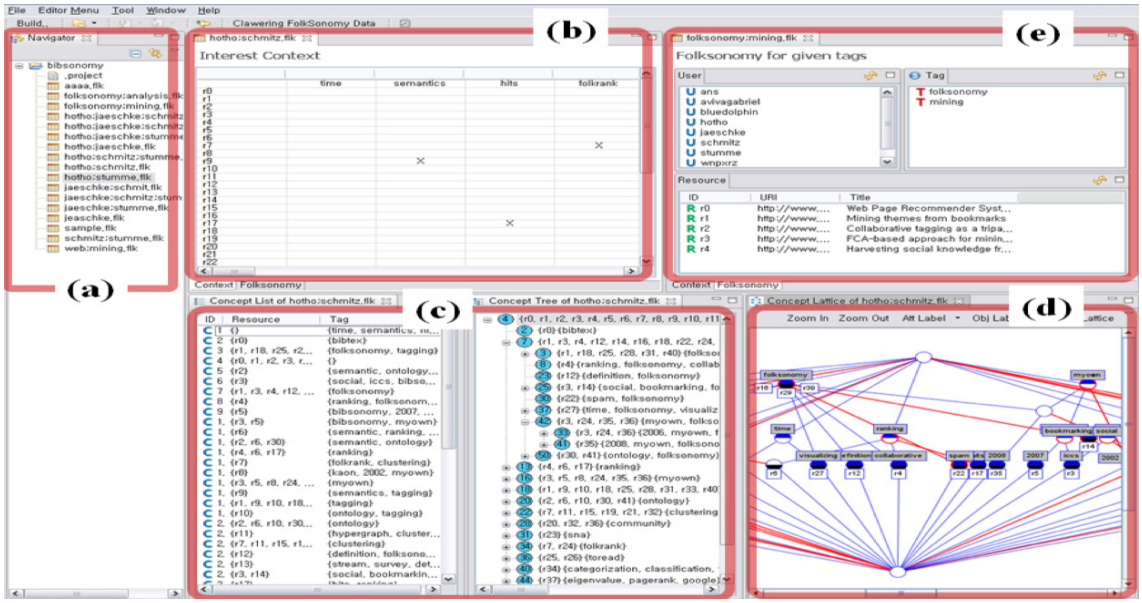


Figure 3.13: The FolksonomyAnalyzer interface with its five views Control View (a), Context View (b), Concept View (c), Concept Lattice View (d) and the Folksonomy View (e).

Folksonomy (CKDF) which is described as an FCA approach of folksonomy mining (cf. [Kang et al., 2009]). In folksonomies, users tag resources, which results in a triadic relation between users, resources and tags. Folksonomies are created in *youTube*[2], *BibSonomy*[3], *Flickr*[4] and *delicious*[5].

Special about CKDF is, that it is concerned with the transformation of the triadic relationship between user, tag and resource into three dyadic formal contexts about just two of the components. Having obtained the formal contexts users-tags, users-resources and tags-resources, a Formal Concept Analysis is conducted on each of them. The example given by the authors examines user communities within delicious.

The Interface of FolkosonomyAnalyzer consists of five major windows, as shown in Figure 3.13. The *Control View* is indicated by (a) and lists all tagging systems the user has access to. The *Folksonomy View* is shown as (e) and shows tags, resources and users belonging together when she selects an instance of these sets. The most interesting part, the FCA View comprised three windows. The Context View (b), shows a simple MS Excel-like *Attribute Matrix* of the context. The *Concept List* view and *Concept Tree* (c) gives an overview of all the existing concepts and finally the Concept Lattice View (d) showing a Concept Lattice created using ConExp.

The concept lattice shows only one of the three formal contexts. But, concepts are used as pivot to move to another formal context. By selecting the intent of a concept a new context is build upon the concept's extend and the corresponding set of objects from the third set. By that way the user can travel along interesting concepts and move between the three contexts user-tags, tags-resources and resources-users.

[2]http://www.youtube.com/
[3]http://www.bibsonomy.org/
[4]http://www.flickr.com/
[5]http://www.delicious.com

The overall focus of FolksonomyAnalyzer is to lower the complexity of the three dimensions of data and the corresponding shift in information and focus. Also it is an application that combined different views on a data set and links them together interactively.

3.3 Integration Approaches

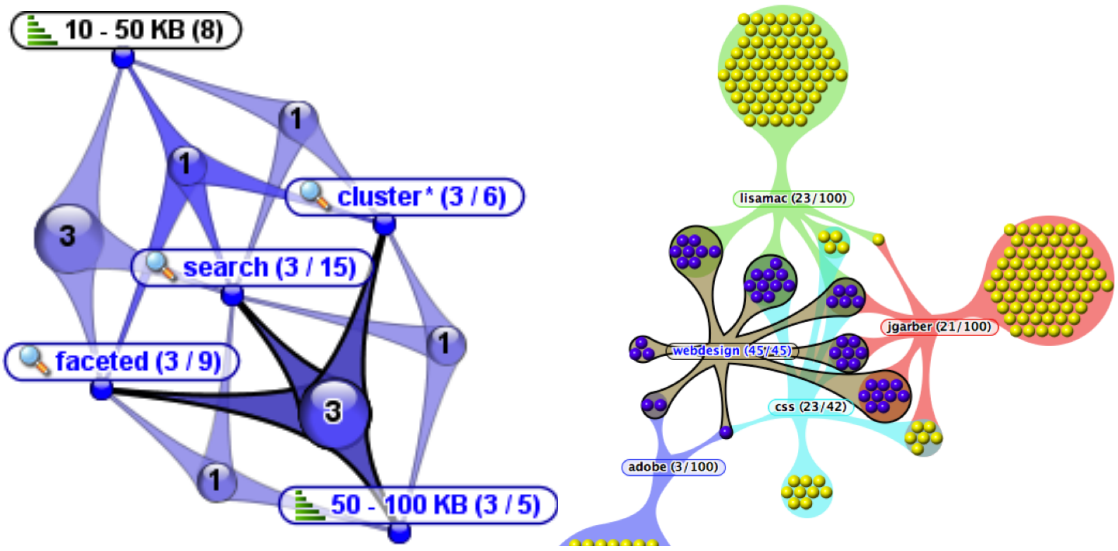
Having presented projects and techniques from faceted navigation and FCA applications, this section presents four projects that use concepts from both, faceted navigation and FCA. The section evaluates the presented approaches under a general scheme to extract similarities, trends and common techniques. Criteria are structure of facet values, presentation of facets, application and visualization of lattices, result representation and how they serve for focalized search, exploratory search and data analysis. Table 3.1 at the end of this section summarizes the results.

Aduna Autofocus is a faceted browser that uses a visualization similar to concept lattices in order to present relations between results of a query (cf. [Hearst, 2009, p.268]). Figure 3.14 shows two types of these *Cluster Maps*. Results are clustered according the values of a faceted search. Facet values are written as labels on the corresponding clusters. Relations between clusters indicate the a certain facet value defines a certain cluster. In this manner, the visualization equals a concept lattice, without possessing a universal or contra-dictionary concept. The right cluster map in Figure 3.14, is an extended version showing the objects within each cluster by yellow bulbs. In this special example the cluster map is used to visualize groups and relations in delicious [Klerkx and Duval, 2007]. In the right example, colors are used to distinguish values and simplify the reading. The example shows intersections of mostly two values and one intersection of three values, while presenting five attributes. Figure 3.14 gives a hint of the problems with the visualization when dealing with many values and interactions.

Values are selected from a facet lists not shown in Figure 3.14. Values are flat instead of ordered into a taxonomy.

FaIR is a system, that scales very well in terms of many facts and large value taxonomies [Priss, 2000a]. In FaIR, concept lattices are used to formalize a facet value hierarchy. Figure 3.15 shows such taxonomy represented by a directed acyclic graph (DAG). It shows the facet value taxonomy of a facet on book contents. The hierarchy is defined as a thesaurus, that is according to the semantics of the terms. It does not represent the actual shape of the data. Objects can be classified under one concept per facet only.

FaIR creates one such graph for each facet and the user can pose queries to the system by selecting concepts from the taxonomy, objects retrieved according to two query types: exclusive and inclusive. Exclusive means that only those objects are retrieved that satisfy only this term , while inclusive retrieves all objects that satisfy at least the search query. Figure 3.15 shows two queries whereby the exclusive query



(a) Aduna Autofocus (b) Cluster Maps

Figure 3.14: Screenshot from Aduna AutoFocus.The right is a screenshot from an application using cluster maps to visualize tag and user relations in Delicious. The current example shows two users `lisamac` in green and `jgaber` in red. The other colored `octopuses` belong to tags, even if not apparent. The blue and yellow balls represent bookmarks collected by users or assigned with tags.

result is shown by a dotted line and the inclusive one by normal lines.

Two tools, *D-Sift* [Ducrou et al., 2005] and *SurfSleuth* [Ducrou and Eklund, 2005], use one single concept lattice to support search and exploration in faceted data. Both are equal in integration degree, visualization and navigation but differ in the structure of the data used. While D-SIFT works with arbitrary facets and value taxonomies, SearchSleuth works with only two facets with eight flat values.

Figure 3.16 shows a screenshot from D-SIFT used with data about mobile phones. The authors do not refer to facets explicitly but group formal attributes of same characteristic. If attributes are numerical, such as `price`, a scaled context is used to divide the attribute into equal intervals. Thus, attribute groups correspond to facets and attribute values are facet values.

D-SIFT allows to build boolean AND and OR queries. To add values that are mandatory for all retrieved objects the are added to the *zoom* set. These attributes are combined by an AND operator. Attributes that are not mandatory are put into *filter* set and are combined by the OR operator.

The lattice contains only objects that apply to all values from the *filter*. The universal concept shows the number of objects that do not apply on any of the filter values, while the bottom concept contains the objects that satisfy all attributes from the filter set. The contra-dictionary concept in the example is empty. The remaining concepts are tradeoffs between the values in the *filter*.

The user can add and remove values to *zoom* and *filter* as well as between them. By

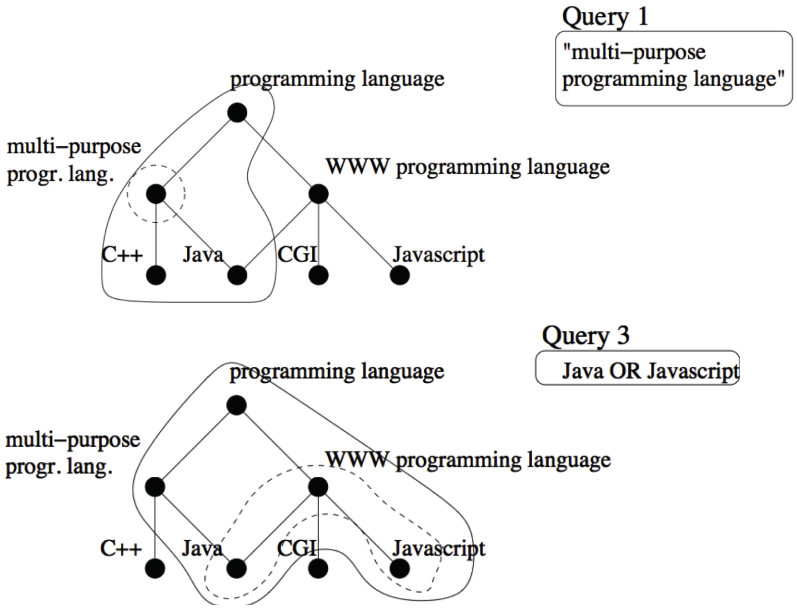


Figure 3.15: Two queries to facet taxonomies in FaIR and their result representation. Black lines designate results of an inclusive query and dotted lines comprise results of an exclusive line.

moving a value from filter to zoom, the lattice gets smaller, since all concepts are omitted where the particular value is not in the intent. Vice versa, the lattice gets larger.

D-SIFT uses faceted data but creates advanced queries than normal faceted navigation. The lattice visualizes the results and alternatives within the results.

3.4 Conclusion

This chapter gave a brief overview of techniques used in faceted browsing for guiding and supporting the user, presented software, browsers and visualizations for formal contexts. Finally projects that can be seen as an integration were described. Advanced techniques for faceted browsing are visualization of query preview, alternative presentation of facets by means of diagrams and maps and direct navigation within clusters of results. All those techniques extend common faceted search to wards exploration and data analysis. Nevertheless, appropriate data visualization techniques are not integrated in those faceted browsers.

FCA bases browsers use common navigation techniques such as tree structures or tag lists. Visualizers employ fisheye, bounding and zoom to keep concept lattices small well arranged. Highlighting of lines shows the origin of a certain concept as well as its children.

Table 3.1 shows a summary comparison of the Cluster Maps, FaIR and D-SIFT. It shows how much of an integration has been done, also non of the tools are developed with that intention. Value taxonomies are managed by FaIR and D-SIFT. Boolean

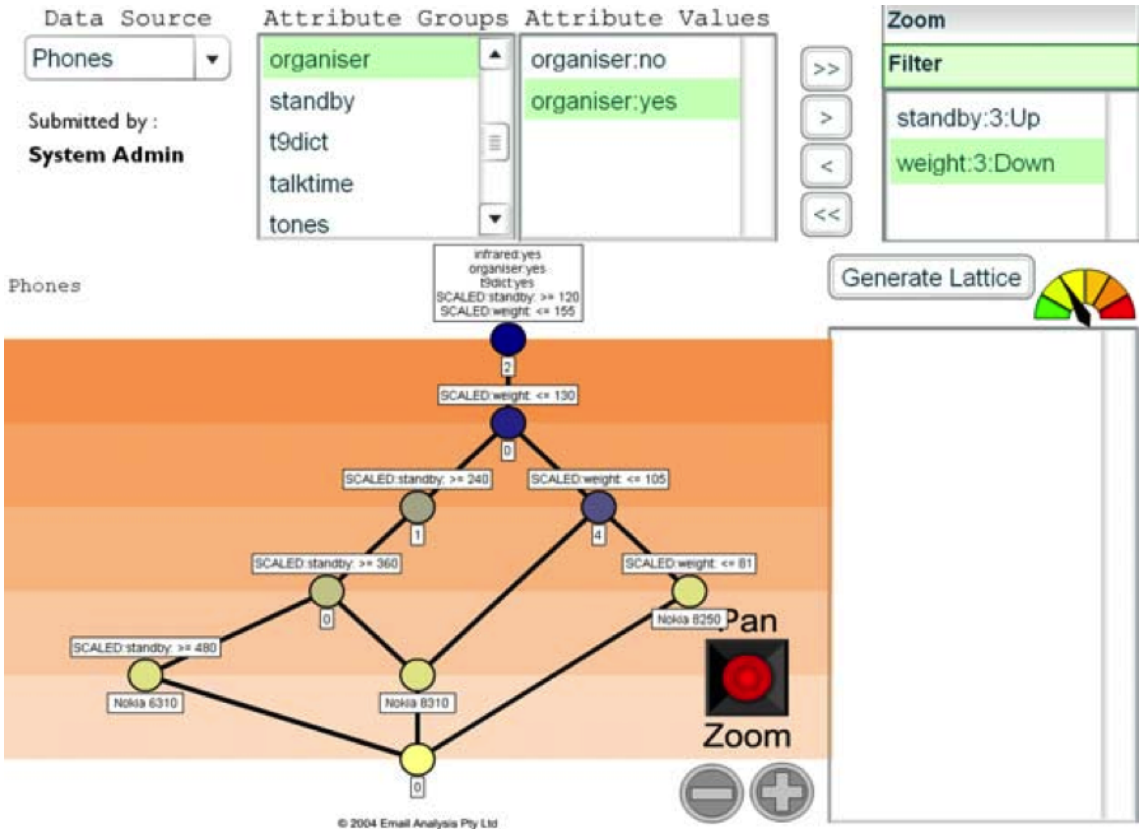


Figure 3.16: Screenshot of D-SIFT showing facets and attrubutes on the top and the concept lattice representation in the center. The lattice shows the formal context defined by selected attributes.

	Aduna/ClusterMaps	FaIR	D-SIFT
Data Structure	Flat values	Value taxonomies	Value taxonomies[6]
Facet presentation	Text list	One DAG for each facet	Text list
Concept Lattice			
Formal context	Lattice like structure	One DAG for each facet	One lattice
Visualization	Colored values, concepts differ in size according extent, relations are "tentacles" indicating direction	DAG, no actual lattice, no bottom concept.	Concepts are colored according lattice level
Navigation	Hover item sets to highlight related ones	Select concepts for specifying query	Pan&Zoom, Hover concepts
Result Representation	Text list	*(FaIR has never been implemented)*	Labeled concepts
Query	AND, OR	AND, OR	AND, OR
Tasks			
Focalized Search	X	X	XX
Exploratory Search	X		XX
Data Analysis			X
Explicit integration	no	yes	no

Table 3.1: Summary on integration approaches for faceted navigation and concept lattices.

queries are supported by the AND and OR operator, allowing to visualize mandatory and additional facet values. Concept lattices are used for displaying those values and allow the user to find tradeoffs between criteria while searching for certain items. However, Cluster Maps do not scale well in terms of many facet values and FaIR does not adapt to implicit structures in data, because the taxonomy is predefined. Both facts lower the limit the possibility of a wider exploration of the data base. D-SIFT is the only tool that makes exploration usable by its *filter* and *zoom* policy. A wider data analysis, however, is limited by missing visualization techniques.

Except from FaIR, which shows the value taxonomy by a graph, facets and values are still periphery to lattices and remain simple text entries and lists.

The next chapter investigates faceted navigation and FCA in detail in order to define potential integration points. The chapter revives examples and techniques explained here.

4 Comparing Faceted Navigation and Formal Concept Analysis

Last chapter showed that approaches were made by faceted browsers to support search and exploration visually. On the contrary, concept lattice representations are applied to other domains than mathematics and used for knowledge retrieval. Integration approaches of faceted navigation and FCA exist, but are not widely used. The last two chapters also mentioned some similarities between faceted browsing and FCA, especially in terms of data structure, querying and object retrieval. This chapter presents a comparison of faceted navigation and FCA in order to point out further similarities and possibilities for an integration.

The main link between faceted navigation and FCA is that both techniques deal with sets of similar items and that their data structure is basically the same. Carpineto states that *"each class of the lattice is seen as a query (i.e., the intent) with its associated set of documents (i.e., the extent)"*[1]. This is true for all conjunctive queries. Nevertheless, he does not refer to further query techniques such as `OR` and `NOT` queries or data structures related to such a query like attribute hierarchies or conceptual graphs.

Despite of many similarities, there is only one comprehensive discussion in literature that opposes faceted navigation and FCA directly (cf. [Sacco and Tzitzikas, 2009, p.59ff]). In their conclusion, the authors mention that in practice, lattices become to big to be useful, whereby Dynamic Taxonomies are used to prune unimportant parts of the concepts and make the search space feasible.

They hold two views. First, they argue in favor of an application of extensions. Various extensions on FCA and faceted search do already exist and have to be compared in terms of a possible combination. It is assumed that extensions from one could already be adapted by the other.

Their second argment is a possible functional combination whereby faceted browsing is used as the front end to the more complex Formal Concept Analysis (cf. [Sacco and Tzitzikas, 2009, p.64ff]).

To approach this thesis of similarity and extension, it is made a more detailed investigation on Faceted Search and FCA. Since this work is primary concerned with a visual and interactive integration, a major focus is on the character of the interface in both technologies.

Because the terminology is different in both domains but same or similar ideas are

[1] [Carpineto and Romano, 1995, p. 94]

described, Table 4.1 refers to the different terms. It builds on the comparison in SACCO 2009a, page 61 and is used as a general reference throughout this thesis.

Faceted Search		Formal Concept Analysis
Object	=	Formal Object
Concept/Facet Value	=	Formal Attribute
Focus	=	Formal Concept
Result Set	=	Concept Extent
AND-Query	=	Concept Intent
Facet	=	Many-valued Attribute
Classification	=	Formal Context
Value Taxonomy	=	Scale Context

Table 4.1: Different Terminology in Formal Concept Analysis and Faceted Search

Objects are the same entities in faceted navigation and FCA, so are facet values and formal attributes. In faceted navigation the notion *concept* is used to describe a single value of the facet and can be assigned to objects. By doing so, the object is explicitly labeled as instance of this *concept*. A *formal concept* in FCA is characterized by the Galois Connection between *intent* and *extent*. Thus, a formal concept has its equivalent in the current *focus* of a faceted search. If the values from the *query* are combined by the AND operator, the query can be seen as intent and defines a *formal concept*. The *extent* of this concept is the *results set*. A *facet* corresponds to a *many-valued attribute* in FCA and a *value taxonomy* is the *Contextual Scale* of the *many-valued attribute*.

The comparison in this chapter starts with a brief overview of the main scopes of faceted navigation and FCA and is followed by an overview over data structures, computation of result sets and concepts as well as navigation methods. The final discussion in this chapter is about posing queries as well as the proposition of a interface design pattern collection for faceted navigation and FCA. The collection gives an overview over the applied techniques and is some patterns are reused later in Facettice.

This section concludes with a description of an integration potential and outlines major integration points that are approached in Facettice.

4.1 Information Retrieval Activities

Faceted Browsing is used for quick object retrieval, FCA is a mathematic analysis technique for knowledge retrieval. They origin from different fields and FCA is used as a data mining technique, working with data about data. Faceted navigation aims at proving a useful and simple interface to the actual data. In terms of the

information retrieval process, faceted navigation covers focalized search and lookup activities as well as some tasks from an exploratory search (cf. Section 2.1.1). FCA is used to perform more complex retrieval tasks to investigate and analyze the data set. While faceted navigation can be performed without prior knowledge, FCA and reading concept lattices needs some training.

4.2 Data Structure

4.2.1 Taxonomies and Conceptual Scales

Both technologies share the same basic data structure consisting of objects described according to a defined terminology. A faceted classification usually consists of facets grouping facet values. Facet values can be arranged into an explicit value taxonomy. A value taxonomy usually is build upon a `is-a` or `is-part` of relation. If an object is classified under a concept, it is also classified under all its parent concept.

A concept lattice contains two implicit hierarchies, an *attribute hierarchy* among attributes and a *hierarchy of concepts*. The attribute hierarchy can be used directly as taxonomy or being compared to an explicit taxonomy in order to discover inconsistencies. The attribute taxonomy is called an *extensional* classification because it is exploits structures within the actual data to create a data model. This approach can be described as *bottom-up* in comparison to the *top-down* approach of the *intentional* classification in faceted data. In a faceted classification, the model is created prior to the classification of objects. Thus, FCA can be used to create a preliminary classification which can be taken and modified as intentional taxonomy for classifying further objects.

Although the hierarchy of concepts is actually the same as the attribute hierarchy, its semantics are different. The concept hierarchy is a hierarchy of anonymous clusters, rather than of designated classes, as it is the case in an attribute hierarchy. Also, the distribution of objects within a concept lattice is *supremum dense* which means that upper concepts represents large groups of items and lower concepts are detailed classification. By observing the concept hierarchy one can describe the behavior of objects, not classes. For example, an objects is part of two major concepts, one with the intent *a* and the other with the intent *b*. The essence of this example is not that there is a small correlation between *a* and *b*, but that there is an individual that belongs to both groups.

Besides the implicit hierarchies in concept lattices, FCA allows *many-valued attributes* (cf. [Ganter and Wille, 1999, p.37ff]). Formal attributes are *one-valued* if they are only *true* or *false* for objects. Many-valued attributes can be compared to facets because they contain several values. The formal attribute `Color`for example, can have the values `blue`, `red` and `yellow`, equally to facets.

Since concepts can only be created from one-valued attributes, many-valued attributes must be transformed. A one-valued attribute is created for every possible

value. This transformation is done by a *conceptual scale*, which is a formal context that defines implications among the new one-valued attributes. It can hence define an intentional hierarchy among those one-valued attributes. For example, the conceptual scale to transform the many-valued attribute `Locality` can has one-valued attributes for continents, countries, regions and cities. These one-valued attributes are considered objects of the conceptual scale as well and the incidence relation indicates whether an attribute implies another. In terms of the locality, certain cities imply certain regions and so forth. The general formal context contains then only one-valued attributes an the implications from the conceptual scales are used to complete the assignation of the newly created one-valued attributes to objects.

4.2.2 Continuous Attribute range

Not only can conceptual scaling be used to model arbitrary facets and value taxonomies in FCA, but also to model attributes with a continuous range such as `price`, for instance. The formal attribute `price` is divided into one-valued attributes that describe a certain range. An object is now assigned a certain range. Due to the possibility of creating hierarchies with conceptual scales, an object which's *price* is 20, is assigned ≥ 10, ≥ 20 as well as ≤ 30 and ≤ 100.

Faceted browsers such as *VisGets* use range slider to specify a range within a continuous facet.

4.2.3 Facet Hierarchies

Section 4.2.1 mentioned a value taxonomy for facets. Within a system of many facets, it is useful to group facets into larger facets. The DelViz Visualization Taxonomy, presented in Chapter 5, groups the facets `Data Type`, `Data Structure` and `Domain` into the super facet `Data` in order to emphasis a distinguishing from facets about visualization and interaction.

There is no implication among the facets like in value taxonomies. An object classified under a value of `Domain` does not imply that it is also classified for `Data Type` and `Data Structure`.

In FCA, conceptual scales are used for modeling attribute taxonomies but can not be used to group many-valued-attributes. Facet hierarchies, however, can be used to reduce and decompose large data sets.

4.3 Computation

The two main problems of large concept lattices are visual clutter and performance of computation (cf. Section 2.3.4). Concept Lattices need to be calculated before the user interacts with them. The same happens if the formal context changes.

Facated Browsing on the other side calculates the results *on-the-fly*, it calculates each result set when the user specifies its query. Lazy computation is a major advantage of faceted browsing systems and allows them to deal with large data bases. If for every facet value the corresponding values are stored or cached the time complexity is $O(|G| * |M|)$ where G are the objects and M are the attributes (cf. [Sacco and Tzitzikas, 2009, p. 63]).

For calculating concepts for FCA, various algorithms exist. Ganter's Algorithm is the first one and has an exponential time complexity $O(|G|^2 * |M|)$ [Ganter and Wille, 1999]. A general comparison of computational complexity of algorithms for generating concept lattices can be found in KUZNETSOV AND OBEDKOV 2002..

A possible tradeoff-between pre calculation and on-the-fly computation are *Iceberg lattices* as explained in Section 2.3.4. They compute only the top most concepts of the lattice according to a threshold of a minimal support. The rest of the lattice can be computed later. *Alpha Galois Lattices* are a similar approach, clustering concepts by similarity [Pernelle and Ventos, 2003]. Concepts are merged if the confidence between them exceeds α.

4.4 Query Construction

In faceted navigation, every combination of facet values is called a *query*. Every FCA concept represents a boolean `AND`-query. Faceted Browsers also allow queries by the `OR` and `NOT` operator (cf. [Ferré, 2009]) although no common used interface solutions exist for that purpose. In comparison to concept lattices, the result presentation is trivial. Priss discusses the idea of calculating formal concepts on the base of `OR` and even `NOT`-operators, but concludes that this approach yields to much concepts (cf. [Priss, 2000b, p.133]).

This paper rises the question if there is no actual need to calculate concepts based on an `OR` or `NOT`-intent, because common concept lattice already show those objects in an implicit manner. For example, the objects in the filter of a concept c with intent m are those objects that correspond to at least one attribute from m. Objects in the extend of concepts that are not sub concepts of the filter of c are objects that represent results of a `NOT`-query in terms of the attributes in m. In that way, a lattice can be used to visualize at least simple queries.

Following De Morgan's Laws one can transform every query $p \vee q$ into $\neg(p \wedge q)$. Thus, the concepts denoted by $p \vee q$ are all concepts that are not descendants of the concept $(\{pq\}', \{pq\})$.

4.5 Navigation

Eklund and Villerd state that a user does not interact with the lattice on the whole but rather navigates from one concept to another, while removing or adding attributes to the current intent (cf. [Eklund and Villerd, 2010, p.305]). Projects such as *The Virtual Museum of the Pacific*, *ImageSleuth* and *CREDO* are good examples for that.

The same holds for faceted navigation, where the user narrows the result set by adding an facet value and enlarges it by removing values. The navigation technique of faceted navigation is the same as for traversing large lattices. Five navigation techniques can be described for faceted navigation and are explained and adapted to navigation within concept lattices in the following (cf. [Stefaner et al., 2009, p.78ff], [Ferré, 2009]).

Zoom-in

Zoom-in means to restrict the search result set by adding more facet values to the constraint set, because the user zooms into the data set. The contrary is *zoom-out* whereby the user removes certain facet values from the constraint and hence widens the result set. In a concept lattice a zoom corresponds to a navigation to a sub concept which works in the same way. In FCA, a zoom operation onto a concept c means to omit all concepts that are not sub concepts of c. *D-SIFT* implements this zoom techniques explicitly.

Shift

A *shift* is performed by exchanging two facet values. It corresponds to a *zoom-out* and *zoom-in* consecutively. The effect is a "horizontal" move in contrast to a vertical zoom. In a concept lattice, a user would navigate to a sibling concept via either a common parent or a common child concept. This principle is implemented in *SearchSleuth* (cf. Section 3.2.2). For example, the current query is `student` and `art history`, the user exchanges `art history` by `computer science`.

Pivot

A dynamic taxonomy shows only those facet values which are still present in the current result set. If a user starts a completely new query based on at least one of those values, he performs a pivot operation. The new query has no common facet value with the old one. The attribute the new query bases on is called *pivot*. In the previous example the attribute `tu-dresden` might occur. If the user selects `tu-dresden` and deselects `student` and `computer science`, `tu-dresden` has become the picot.

Performing a pivot operation in a concept lattice corresponds to navigate from a concept c_1 to a concept c_2 so that the join of c_1 and c_2 is the universal concept. In the example, the intersection of the two queries is empty.

Slice and Dice

Removing the facet value v_1 of facet f_1 from the current query and adding a value v_2 of a facet f_2 results in a *slice and dice operation*. The example of `student` and `computer science` could be changed to `student` and `dresden`. By shifting from the facet `studies` to the facet `location`, the user changes the semantic of this query, in contrast to the navigation step *pivot*.

Query-by-Example

A *query-by-example* is the retrieval of objects that are similar to a given set of objects. All their facet values together create a new `AND`-query. By two students, one that studies `art history` and the other one `computer science`, results in the query `is:student AND studies:art history AND studies:computer science`. The new result set might be very small.

In a concept lattice the result concept of an query-by-example is the *meet* of the most specific concepts the objects belong to (cf. Section 2.3.2).

The relation between navigating among concepts in a concept lattice and changing the query in faceted navigation is evident. Each of the presented navigation techniques performed in faceted navigation has its correspondent in a concept lattice of the same data set. Moreover, faceted navigation is navigating through concept lattices.

4.6 Interface Design Patterns

Design patterns are a common way to describe standard solutions for problems in a certain domain. An essential high-level overview to *User Interface Design Patterns* was made by TIDWELL 1999 and TIDWELL 2005, but many others exist [Welie, 2010, HCI, 2010, Erikson, 2010, Fincher, 2006, Fincher, 2009].

Patterns for Information Design are a collection within a greater collection of Interface Design Patterns [Potsdam]. These *InfoViz* patterns are divided into the categories *Display Patterns*, *Behaviour Patterns* and *Navigation Patterns*.

Design patters for faceted navigation are described in SACCO AT AL. 2009 p.86ff and are subsumed here in short.

- *Selection Management* describe patterns such as *check boxes* and *range sliders* to select facet values.
- *Revealing Hierarchy* for giving an overview on facet and value hierarchies. Solutions are simple *explorer trees*, *collapsable facet panels* and *continuos zooming* (cf. [Dachselt et al., 2008]).
- *Facet Management* patterns deal with the problem to show all facets on the screen and distinguish facets by colors for example.

- An *History Navigation* pattern are *breadcrumbs* which allow the user to go back in history and see her actual focus.
- Patterns for *Visualizing Properties* are *numbers* on values *Elastic Lists* or bar charts as in *RelationBrowser++*.

Based on the observations of this chapter, the chapter on related work and especially Table 3.1, this section creates a taxonomy of concept lattice design patterns (CLDP) for concept lattice visualizations. It classifies patterns into Perspective Patterns, Visualization Patterns and Navigation Patterns. Since there is no comparable collection of CLDPs, this taxonomy presents a first approach. Each pattern comprises a name, a problem description a solution and references to related patterns.

The pattern collection serves as an analytical method to classify and compare CLDPs as well as to give the opportunity to identify extension points. A second purpose is to provide independent and purposeful building blocks for further visualizations and tools.

4.6.1 Perspective Patterns

Mining patterns do not directly belong to the core FCA functions but apply algorithms to re-arrange or reduce a concept lattice.

Iceberg Lattice

Problem: Large Context
Solution: Calculate only concepts c whose support $supp(c)$ is more than $s \in [0, 1]$
Result: Only the top most concepts will be displayed. By decreasing s, more concepts will be shown. If $s = 0$, the whole lattice with bottom concept is shown

Alpha Galois Lattice

Problem: Large Context
Solution: Cluster concepts by similarity $\alpha \in [0, 1]$
Result: If α is 0, only one concept is shown, if α becomes larger more concepts are shown until the complete lattice is shown for $\alpha = 1$

Attribute hierarchy

Problem: Generality relations between attributes
Solution: Lattice is interpreted as hierarchy among attribute concepts
Result: A directed acyclic graph

Concept Tree

Problem: Large lattice or no resources for showing lattice but necessity to traverse lattice in an easy manner
Solution: Create tree with node duplication for lattice starting with top concepts
Result: Navigable tree structure that can be visualized by simple tree lists or tree maps
Exmaples: CREDO

4.6.2 Visualization Patterns

Visualization Patterns aim in increasing the readability of concept lattices without changing the structure or data as Filter patters do.

Reduced Labeling

Problem: Concept lattice with many attributes and objects
Solution: Show attribute and object names only at corresponding attribute or object concepts
Result: Every attribute and every object name appears only once within the lattice
Examples: Toscana, D-SIFT, MailSleuth, SurfSleuth, Ulysses

Nested Line Diagram

Problem: Large context
Solution: Group attributes and create a concept lattice l_{outer} for one attribute group. For each concept in l_{outer} create a lattice l_{inner}
Result: The big lattice l_{outer} shows for every concept a specific lattice l_{inner}
Examples: Toscana

Fisheye

Problem: Large concept lattice
Solution: Reduce size of concepts depending on their to the current focus, possibility of omitting very far concepts
Result: Only the direct neighborhood of the current lattice is visible. If the user changes its focus, the visible neighborhood
changes as well.
Examples: Ulysses

Distinguish Top and Bottom Concepts

Problem: Increase usability for users without background in FCA
Solution: Distinguish top and bottom concepts visually from other concepts
Result: Emphasizing top and bottom concept as well as direction of concept lattice.
Examples: MailSleuth

Layering

Problem: Increase usability for users without background in FCA
Solution: Layer background of lattice in order to emphasize direction or chose layered layout approach
Result: Implicit grouping of concepts of same depth
Examples: MailSleuth, SurfSleuth

Line Highlighting

Problem: Show relevant parent and child concepts
Solution: Highlight all concepts and relations until top and bottom concept
Result: Users see particular attribute hierarchy section of the concept lattice
Examples: ConExp

4.6.3 Navigation Patterns

Navigation patterns aim a navigation within the lattice, starting at a particular concept and defining the way the user can take. Besides the already mentioned patterns *Zoom*, *Pivot* and *Shift* following are part of this collection.

Concept Neighborhood

Problem: Retrieve similar objects or similar attributes
Solution: Traverse concept neighborhood iteratively.
Result: If lattice is already calculated, this is a fast and solid method to retrieve all similar items ordered by similarity.
Examples: ImageSleuth

Bounding

Problem: Bound search or navigation space within lattice
Solution: Restrict direction in lattice to only or combinations of parent concepts, child, concepts, left-sibling and right-sibling.
Examples: Ulysses

4.7 Integration Aspects

Table 4.2 shows a summary of the comparison of faceted navigation and FCA.

The data structure used for faceted classification and FCA is basically the same. Also FCA ca handle attribute taxonomies by conceptual scales. A dynamic taxonomy is represented by the intent of child concepts in a concept lattice.

Facets, a facet hierarchy and value hierarchies could be used to reduce a formal context, define precise foci, define levels of abstraction and examine facet characteristics.

In terms of a value/attribute taxonomy, a faceted classification is a top-down approach because the taxonomy is created prior to classification. Besides conceptual scales, an attribute taxonomy is created after classification of the objects. Thus, FCA presents a bottom-up approach to taxonomies, based on the actual instantiation of elements.

An integration in terms of taxonomies helps to reveal inferences among values of the explicit value taxonomy and improve it.

While faceted navigation computes results for a query set on-the-fly, FCA needs to calculate all concepts first before they can be accessed by a user. By navigation from result to result, the user navigates through the concept lattice. Navigation modes in faceted navigation are easily represented in concept lattices.

One possible integration is the successive construction of the lattice by user navigation. Instead of creating the whole lattice at once, it is created upon and adapted to the users search history. A lattice is able to show parts of the data set the user has already visited, provide alternatives by context neighborhood and visualize precise navigation paths. The lattice can become the map of the search space, adapted to the users view.

When the user poses queries with an `OR` or `NOT` operator, his navigation paths become complex. Nevertheless, it must be investigated how a concept lattice can visualize complex queries.

The interface of faceted browsers and concept lattice application differ most. Each technology established certain interface design patterns that are widely reused. The tightest integration made in D-SIFT, uses only the Facet list pattern from faceted browsers.

A simple combination of interface patterns from faceted navigation and FCA is a first step. But, the creation of a holistic interface is different.

Faceted navigation hides complexity of data by accessing them by means of the data model, which are the facets and their values. The trend in faceted browsers is towards involving information from actual data to create dynamic taxonomies and query previews. FCA, in contrast, shows complexity and instantiation of data in order to create a general model an understanding for data.

Not much information visualization techniques have been adapted to concept lattices.

Property	Faceted Search	Formal Concept Analysis
Information Retrieval	Focalized Search , Exploration	Data Analysis, Exploration
	Known Item Search, Exploratory Search	Exploratory Data Analysis
	Object-Seeking, Knowledge Seeking	Knowledge Seeking, Wisdom-Seeking
	Navigation, Browsing	
Data Structure	Explicit facet and value taxonomy	Explicit conceptual scales
	Dynamic Taxonomies	Implicit attribute and concept hierarchy
	Thesaurus	
Computation	Lazy computation	Pre-calculation
	Linear time complexity	Exponential time complexity
Query	AND, OR, NOT	AND, OR, NOT (cf. Section 6.3.3)
Navigation	Zoom, Shift, Pivot, Slice and Dice, Query-by-example	
Interface	Facet list, tree, tree map	Bounding
	Breadcrumbs	Line highlighting
	Facet diagrams	Attribute Hierarchy
	Color coded facets	Concept Tree
	Query Preview	Reduced Labeling
	Rank results	Nested Line Diagram
	Range slider	Fish eye
	Set based browsing	Distinguish top and bottom concept
		Layering
		Iceberg Lattices, Alpha Galois Lattics

Table 4.2: Overview of the comparison of faceted navigation and Formal Concept Analysis.

In order to analyze data, patterns must be described which can be emphasized visually. Therefore, the next chapter presents a real-world data set. It is accessed by average users for search and browsing as well as by experts for data analysis. The mentioned integration points are approached in Chapter 6 and are:

1. Information visualization within concept lattices
2. Query preview
3. Development of an interface component for direct navigation inside concept lattices
4. History visualization and iterative lattice construction

5 Application Domain and Task Analysis

This Section describes and analyzes a data set that serves as a primary reference for the concept and realization of Facettice. Two associated visualization projects are presented and usage tasks as well as additional requirements for an interactive search and analysis tool are presented.

5.1 DelViz Visualization Taxonomy

The *DelViz Visualization Taxonomy* collects and classifies visualization projects (cf. [Keck et al., 2010]). The data are bookmarks stored on *delicious*[1] and tagged with terms from the visualization taxonomy.

5.1.1 Metrics

The metrics in this section are retrieved from the data set stored at Delicious and varies in its extend from the taxonomy as presentet in Figure 5.1. Nevertheless, it gives an good overview over the data set.

Figure 5.1 shows the taxonomy consisting of three major facets `Data`, `Visualization` and `Interaction`. Each of them has three or six sub facets with an average of 4.5 facet values. The taxonomy comprises 63 facet values assigned to 716 bookmarks with an average tag assignment of 9.9 per bookmark. Because this last number is quite large, high interferences among the facet values occur. The corresponding lattice contains many concepts and relations. The total amount of assignments from tags to bookmarks is 7063. In terms of a maximum amount of 45.108[2] assignments this is a rate of 6.35%. Table 5.1 shows an overview of the metrics collected in this context.

The formal context spanned by these data is according to the facet values, a larger medium-sized context. Examples in books use 5-10 attributes while real-world contexts comprise up to 100. The amount of objects has a minor influence on the size of the concept lattice, since the amount of possible intents, and hence concepts, depends on the combination of attributes.

[1] http://www.delicious.com
[2] 63 facet values $\times$ 716 bookmarks = 45.108

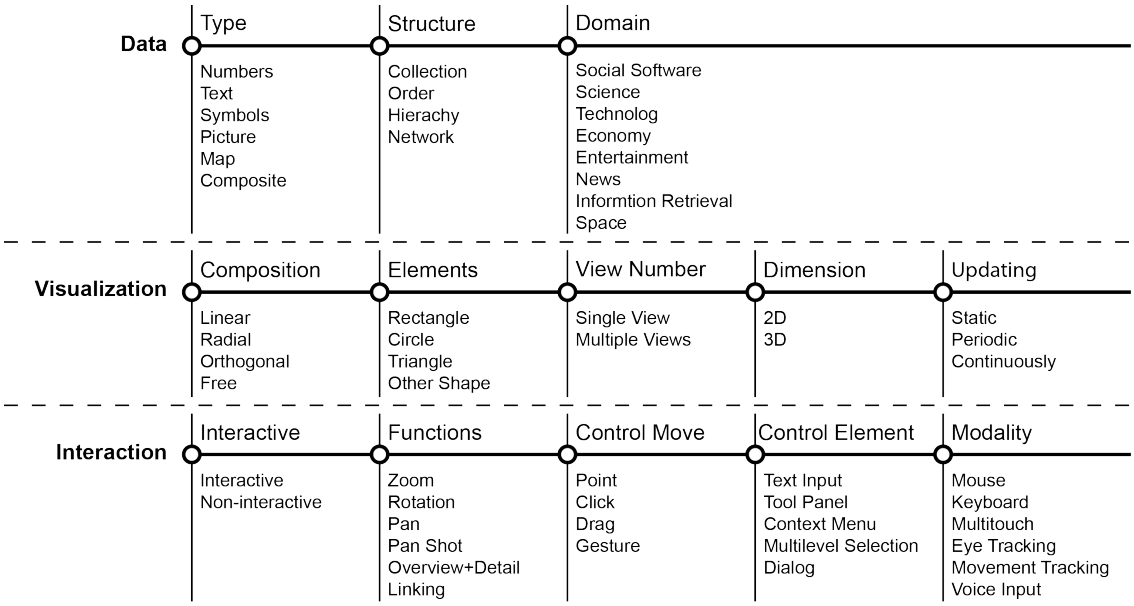


Figure 5.1: Taxonomy of the DelViz Visualization Taxonomy.

Metric	Value
Total amount of facets	14
Hierarchy Depth of facet hierarchy	2
Hierarchy Depth of value hierarchy	1
Total amount of facet values	63
Average amount of values per facet	4.5
Total amount of posts	716
Total amount of tag assignments	7063
Average tags per post	9.9

Table 5.1: Important data metrics of the context

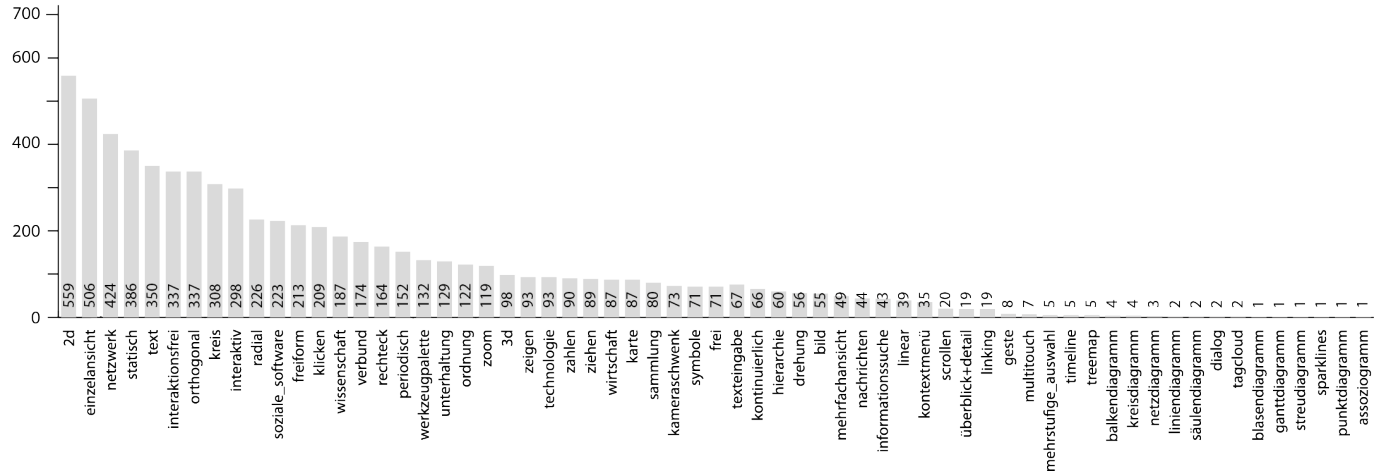


Figure 5.2: The distribution of tags on the bookmarks show a *long tail* distribution (11. November 2010).

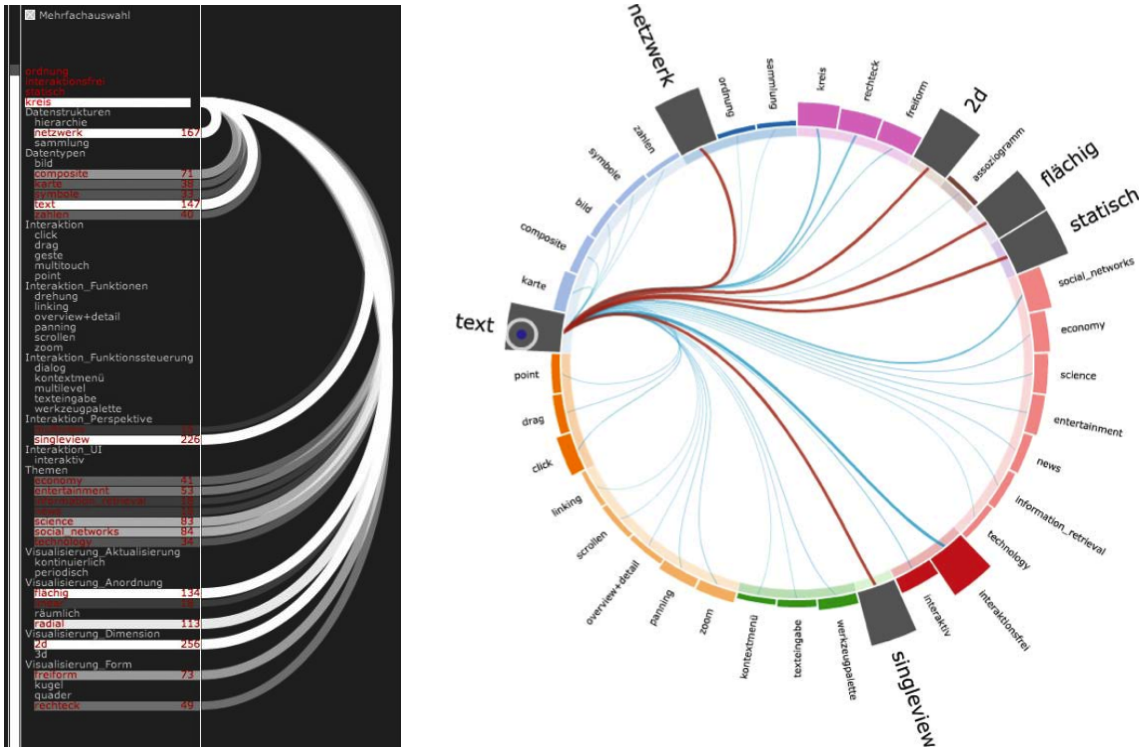
Another aspect of the data is shown in Figure 5.2 which displays the distribution of facet values from the *DelViz Visualization Taxonomy*. The image reveals the *long tail* character of the data (cf. [Anderson, 2006]). The most assigned attribute 2d is used 559 times on 718 objects which is 77%, or a support of 0.77. According to this long tail character, it is likely to yield a distribution of attribute concepts over the whole lattice.

5.1.2 Related Work

There are two tools for exploring the visualization taxonomy and to reveal relations among tags: *DelViz* (cf. [Keck et al., 2010]) and a multi touch table visualization for tag networks (cf. [de Almeida Madeira Clemente, 2010]). Both projects show relations between the taxonomy values based on a similarity measure. The similarity of two tags is calculated by the number of common objects.

DelViz uses a tree composition for the facet and attribute taxonomy and relation arcs to relate similar tags, as shown in the left screenshot of Figure 5.3. The similarity is codified by the lightness of the arcs; the brighter an arc is, the more do these tags co-occur. When the users selects a tag, arcs to related tags are shown, otherwise they are hidden. This techniques is used to explore correlation of tags. *DelViz* also allows browsing the visualization projects. Projects are retrieved by faceted navigation on the taxonomy and are shown on a separate interface component providing screenshots and a detailed description.

The right screenshot in Figure 5.3 shows de Almeida's circular composition for facet values and facet membership indicated by colors. A major advantage of the circular layout of elements is the ability to show relations between all elements in an clear way and avoid node occlusion (cf. [Gansner and Koren, 2007]). Edge bundling was applied by de Almeida to reduce visual clutter and better show trends and clusters within the network (cf. [Holten, 2006]). In addition to that, is is differentiated into



(a) DelViz (b) Tag network visualization

Figure 5.3: Screenshot from DelViz on the left, showing the related tags by thread arcs. The right is a screenshot from the tag network visualization by DE ALMEIDA 2010.

three types of relations indicated by color. Similarity between attributes, is codified by the lightness of the relation line as well as in the height of bars corresponding to circle segments.

When a user selects a facet value, an adapted image is shown, equal to a dynamic taxonomy. The new circle is shown inside the other one. Similar to tree rings, this technique visualizes history and the user can go back. Also, the proportionality between tag occurrence in the last and the current focus is indicated.

The representations of both *DelViz* and de Almeida emphasize an exploration by focused views one attribute. While *DelViz* supports browsing and search, de Almeida allows deeper exploration and analysis.

5.2 Usage Tasks

Discussions on how to generally structure tasks and actions in data analysis and to divide them into smaller graspable actions have been made in general (cf. [Bertin, 1973], [Shneiderman, 1996], [Amar et al., 2005], [Andrienko and Andrienko, 2006], [Zhou and Feiner, 1998]).

YI 2007 analyses various Usage and Interaction taxonomies in the field of Information Visualization. The authors conclude seven own major tasks that are references in the following.

- Select: mark something as interesting
- Explore: show me something else
- Reconfigure: Show me a different arrangement
- Encode: Show me a different representation
- Abstract: Show me more or less detail
- Filter: Show me something conditionally
- Connect: Show me related items

This Section defines three usage scenarios for the visualization taxonomy which has also been reported in [de Almeida Madeira Clemente, 2010].

5.2.1 Focalized Search

Visualization projects should be found by characteristics and for different purposes. Since the objects in the visualization taxonomy are classified under several facets, they can be used to retrieve the objects. An example would be to find *"all interactive science news visualizations that make usage of the map metaphor"*.

The basic challenge for an interface is to provide an overview over the facets and let the user observe the objects in detail. The process of a focalized search has a few

steps as explained in Section 2.1.2. Information, necessary to solve such tasks and the resulting conclusions for a visualization are:

1. provide an overview over *facets, attributes* and their *hierarchy* so that the user knows how he can specify his query,
2. present the search results and let user observe them by detail, and
3. support the user in deciding whether the retrieved results are accurate. If not, enable him to either rewrite his query or retrieve similar objects.

5.2.2 Browsing and Exploration

An example for exploration is to find related projects for a specific visualization technique or look for ideas for a certain domain (cf. [Keck et al., 2010]).

Since this question cannot be expressed by one single Boolean query, it has to be subdivided into several. An initial query could ask for the particular visualization technique. If the size if the result set is to large, the user can chose a further facet value. He would chose the one that yields the most interesting projects. So, the user starts browsing and explores the data set by following interesting facet values.

Conclusions for Facettice are:

1. provide an overview of associated terms,
2. suggest what the next query can be about,
3. support *query preview* so that the user can base her "selecting decision" upon
4. provide a *search history* that gives the user a picture if his browsing behavior and enables him to go back to a certain point in time,
5. support *query rewriting*, which means to change the current query as well as pose a new one,
6. give a *summary of search process* that allows the user to decide whether she has seen every interesting, and
7. attract *interest* and arouse *curiosity*.

5.2.3 Data Analysis

The third scenario describes tasks for data analysis. Analysis tasks require explicit questions which then defines the particular analysis method (cf. Section 2.1.3, [Keim et al., 2008]).

AMAR ET AL. 2005 define low-level tasks of data analysis such as *filter, find extremum, retrieve value, sort, characterize distribution, find anomalies, cluster, correlate, scan, set operations*. Based on AMAR ET AL. 2005, LEE ET AL. 2006 define graph specific tasks divided classified into *Topology-based, attribute-based, browsing* and *overview*.

Since concept lattices are graphs, these tasks are of particular interest. Topology

based means to observe the relations among concepts, central concepts, outliers as well as clusters. Those tasks are close related to overview tasks but the latter implies an holistic picture of the data and its parts. By attribute tasks the authors refer to particular values of network nodes defined by the data model. Browsing tasks have been described in the previous section. Since the described tasks are defined for networks they are taken as an reference for concept lattices.

Concerning an analysis of the visualization taxonomy, there are four main questions which are: (1) Detect clusters and outlier concepts by evaluating correlation and implications among attributes, (2) compare facets by their realization ,(3) test the completeness of the current taxonomy and (4) observe the influence of arbitrary values onto a set of objects.

It might be interesting to *"estimate the distribution of primary visualization techniques depending on the usage domain"*. An answer may reveal that there is a preference for `maps` among the domains of `science` and `news`. Or, that there is preferred technique for each domain. According to the answer, visualization techniques could be grouped by usage domain, data structure or popularity. In contrast to exploration and search, the focus of an analysis is less on the objects but on the attributes.

According the above mentioned four major questions, Facettice needs to

- provide a particular analytical view on each facet,
- analyse facets in parallel,
- analyse facets in combination,
- combine arbitrary facet values, and
- apply information visualization techniques to concept lattices.

In addition to the tasks described in this chapter, an browsing and analysis interface for the DelViz Visualization Taxonomy needs to fulfill three additional criteria. First, the visualization must obtain data from the website directly in order to allow changes on the tags and bookmarks. Second, by the same reason, the application must not limit the number of objects, facets or tags. The last criterion is that a thumbnail image and a detailed description of each object must be shown. The latter one is not a generic solution but Facettice is tested with the DelViz Taxonomy for the present.

6 Facettice

Chapter 4 contrasted faceted navigation and FCA, described similarities and differences between both and pointed out major integration aspects. The chapter also defined a collection of interface design patterns for concept lattices which are partially reused in Facettice. Chapter 5 described a multifaceted data set with three usage scenarios Focalized Search, Exploration and Data Analysis.

The conceptualization in this chapter is divided into four steps. The first examines important data components and describes their relationships precisely. Therefore, a model called *Ontological Space of Data* is defined. The model helps making decisions for visualizing structured and semi-structured data. It is also used to investigate further relationships between a faceted classification and concept lattices. In a second step, the analysis of the Ontological Space of Data leads to the identification of groups of data items and entails conditions for their visualization. The third step describes an interface metaphor in order to generate a consistent interface idea for all components. Based on that, those interface components are related interactively. The fourth step approaches the integration points defined in Chapter 4.

Facettice is a tool that supports faceted navigation by interactive concept lattices. It presents two types of concept lattice visualizations based on a faceted classification and supports faceted navigation in two different ways. The visualizations are described in the last two sections of this chapter, and demonstrated in the next chapter.

Since this chapter describes an integration of faceted navigation and concept lattices, the terms *facet value* and *(formal) attributes* are used as synonyms. *Context* refers to the neighborhood of a concept, while *formal context* means the formal context.

6.1 The Ontological Space of Data

6.1.1 Purposes and Process

The *Ontological Space of Data (OSD)* has three purposes which are at the same time considered steps in a process. The model classifies different data *components*. Components are semantically different parts of the data such as data data model, instances, literals and relations among instances. The term *ontological* refers to the pretension of describing structures and entities of a reality in order to discuss and relate them in a new way.

The OSD does not refer to ontologies as data structure in particular, rather it tends

to describe all types of structured and semi-structured data. Also non-structured data such as text and language, can be analyzed by the model but the results are less substantial.

The second purpose of the OSD is to reveal relationships between the data components. Those relationships can be *explicit* and pre-defined like a class hierarchy or emerge *implicitly* from the OSD. Explicit relations depend on the particular data set but implicit are similar in all data sets, as explained later in this section.

After data components and their relationships are defined, views on the data can be derived for each component, considering its particular character as well as explicit and implicit relations. A view can be simple lists and tables or more complex visualizations. According to the relations among the data components, navigation and interaction techniques between the views can be defined. These techniques serve to tie different views together, transform one view directly into another, nest views and so forth. Interaction between views serves for combining different visual representations of different data components, such as view of the data model and one of the instances. The selection of views, navigation and interaction techniques depends on the domain of the data and specific methodological tasks. Components and relations can be weighted and prioritized.

Not all discovered components and relations need to be directly visualized or shown in a final interface, but every absence must be justified and done by purpose.

For example, a data set consists of students each having an average grade, a matriculation number and a set of visited courses. Components of these data are students, names, grades, matriculation numbers and visited courses. Explicit relations are `has-name`, `has-mark`, `has-number` and `visited-course`. Implicit relations exist between the class of all students and the single individual, as well as between the student and his courses.

A ranking list for a scholarship only considers the mark and the matriculation number. In addition, only the 10 best students should be displayed. The limitation to 10 expresses an implicit relationship between the class of all students and the single individual. An interactive application for stuffs would consider names and visited courses as secondary and hence hide this information from the interface. This detailed information can be retrieved later.

The ranking aspect of the students can be visualized by an ordered list, showing grade and matriculation number. To guarantee anonymity, names must not be shown in this application. Since the example is far simply and staffs are familiar with the data set, it is also irrelevant to visualize explicit relations that students have an average grade and so on. It suffices to label the head of the resulting table.

Classifying and analyzing complex data structures such as faceted classifications and ontologies (cf. [Gruber, 2009]) requires a detailed description of the OSD. In the following the model is developed with respect to faceted data and concept lattices

The OSD is a two dimensional space, spanned by the *Dimension of Scale* and the *Dimension of Abstraction* as illustrated in Figure 6.1. Data components are classified

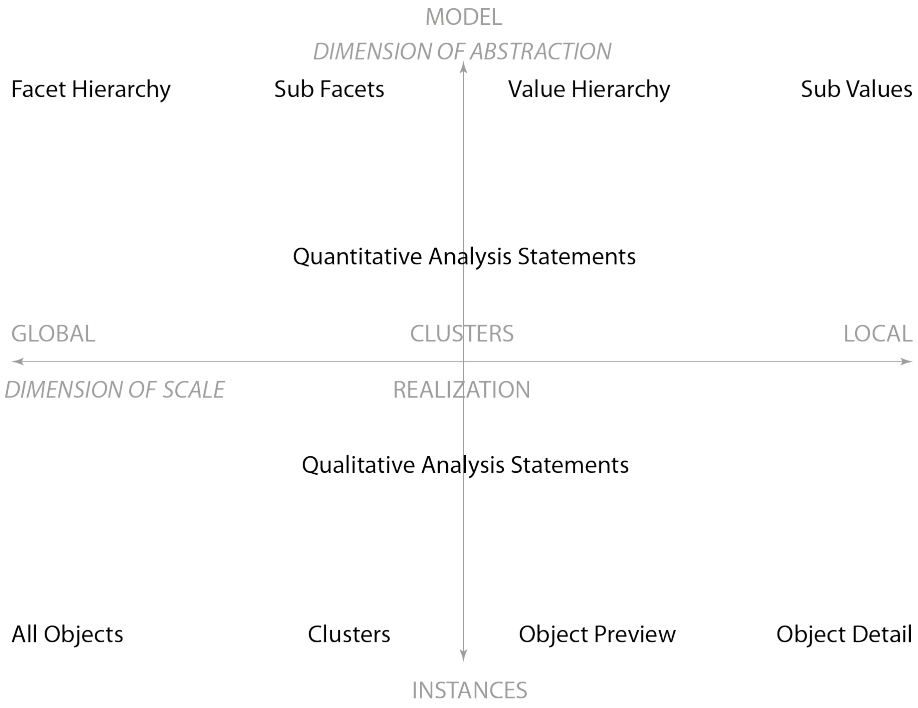


Figure 6.1: The Ontological Space of Data spanned by the Dimension of Scale and the Dimension of Abstraction. In this example, data components from faceted data are classified according these two dimensions.

according to these dimensions. The components of a faceted classification are *facets, facet values, objects* and *object descriptions*. Explicit relations exist between facets and values in form of a hierarchy. Usually there are no explicit relations among objects, as it is the case in instances in ontologies.

6.1.2 Dimension of Scale

Components of a data set can be considered on the whole, from a global point of view, as well as in detail observing single data entries only. This *Dimension of Scale* can be divided into the three segments *global*, *clusters* and *local*. In Figure 6.1 the Dimension of Scale is represented by the horizontal axis. Segment *global* refers to whole data set, *clusters* refer to groups and the immediate context of a particular item, and *local* refers to the description of a single item or literal.

A transition from *global* to *local* is seamless by definition and zooming is the common physical analogy. A transition can also be discrete, for example, when dividing clusters into sub clusters, or when obtaining additional information to an object.

An adaption of a faceted classification to the Dimension of Scale is easy. Starting a faceted search, one yields many objects, then they are filtered according the facets and finally objects can be observed in detail. Most faceted browsers do not show all details of an object but short summaries. Detailed information about objects are URLs, names, descriptions and images; literal resources that belong only to the

particular object.

6.1.3 Dimension of Abstraction

Independently from the Dimension of Scale, every structured or semi-structured data set divides into *model* and *instances*. Facets, facet values and their hierarchy represents a data model for objects. This model defines an abstract framework by that instances are described. Usually, the model is designed by hand, while instances can be retrieved and classified by algorithms.

The *Dimension of Abstraction* ranges from the abstract *model*, such as classes, facets and attributes, to the actual *instances*. The border between *model* and *instances* is fairly strict, in contrast to the seamless transition within the Dimension of Scale. Each data component does either belong to the model or the instances, although a classification is not always trivial.

The data model includes all parts of the data set that are descriptive. They are comparable to columns in a table. Instances correspond to rows in a table. The table analogy is similar to an attribute matrix in FCA where objects are rows and attributes are columns.

Ontologies in computer science make an explicit differentiation into model and instances. The model is called *TBox* and defines the terminological components such as classes, relations, properties and semantics (cf. [Gruber, 2009]). An ontology is *populated* if there are instances within the *assertion box (ABox)* that correspond to the model.

Nevertheless, a strength of the OSD is that it defines a seamless transition between *model* and *instances*, in analogy to the Dimension of Scale. The aspect that closes the gap between the two poles *model* and *instances* is the *realization* of the model in the reality.

The *realization* describes occurrence and shaping of particular parts of the model, according to a certain reality. It can be compared to entries in a table, or the incidence relation in a formal context. By defining *realization*, the Dimension of Abstraction divides into the segments *model*, *realization* and *instances*. This partitioning creates a seamless transition, as in the Dimension of Scale, feasible.

With regards to faceted data, the realization is about the extent of facet values as it is the case in dynamic taxonomies. The issue of a *realization* can be and is described in the next section. Figure 6.1 shows the OSD and data components for faceted data.

The Facet Hierarchy is shown in the upper left corner because it represents a global view on the data model. Following the Dimension of Scale, facets divide into sub facets, values and sub values. The instance set in a faceted classification divides into clusters, object previews and details of an object.

The realization of the data model is described in the next section.

6.1.4 Realization and Analysis Statements

Realization has been described as relationship between model and instances. For the *local* part of the OSD this relationship is described by the assignment of a single object to the data model, such as classification or specification of numerical and nominal attributes. For the *global* part, the sum of these assertions is considered. The result are analytical observations about the realization of the model within a certain reality.

The realization segment can be further divided into *qualitative analysis* and *quantitative analysis*, whereby the first is located closer to the instances and the latter is closer to the model (cf. Figure 6.1).

Quantitative Analysis

The upper level, closer to the *model* describes quantitative information of the model such as extent of a class, occurence of a certain value or existence of relations. These observations are called *quantitative analysis* because the realization of the model is described by numbers of extents.

In facet navigation, these quantities are shown by the numbers for query preview within a dynamic taxonomy or by direct visual variables (cf. Section 3). As explained in Section 4.6, those numbers are a common design pattern in faceted browser interfaces.

Quantitative analysis helps to reveal a primary picture of the realization of the model in a particular data set. It is a mono-directional way of analyzing the data set, because it only considers structure and components of the model, rather than real world structures.

Qualitative Analysis

Another level of analysis considers implicit interrelations within the data set extending or altering the data model. In the worst, such an analysis reveals a violation of structures of the model. A violation occurs if an instance contradicts the semantics of the model, for example, if an object is classified under facet values that are defined exclusive. Usually facets are not exclusive or define equal restrictions, except the classification originates from an ontology [Ion,]. However, if the data model does not define explicit semantics, a real violation of the data model can often only seen by humans. In that terms a visualization would show must show those exceptions.

The data model can be altered or changed if new relations or components exist among the instances. Within a data base about furniture for example, there is a facet `type` with the values `bed` and `sofa`. A quantitative analysis revels the extent of both values. But, only a qualitative analysis can reveal that there are many items that are beds as well as sofas. As a conclusion, an additional facet value `bed-sofa` should be added. Changing or narrowing the set of instances changes the information

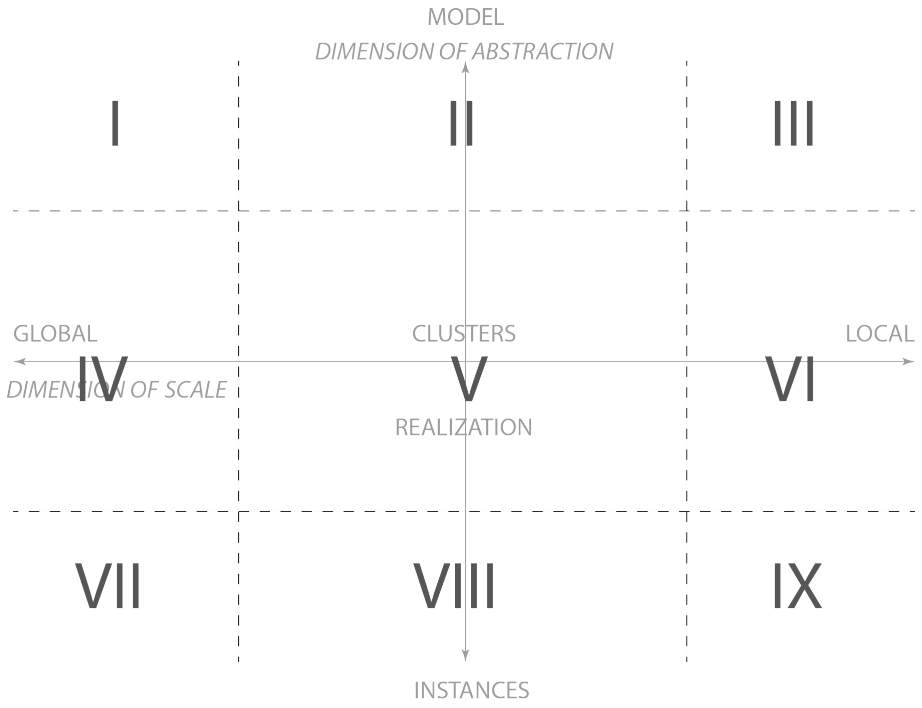


Figure 6.2: Model of the Ontological Space of Data with the nine regions.

from a qualitative analysis, because the set of instances is different. In terms of the furniture example, an analysis yields different results if only living room furnitures are considered.

Formal Concept Analysis enables such a qualitative analysis. Concepts rely on instances that are already classified by formal attributes. Based on that, FCA builds a meta-classification in the form of a concept lattice. The concept lattice can be seen as a particular data model. This schema may differ from the explicit hierarchical data model in faceted classification. Moreover, it reflects a particular reality. The application of FCA in ontology class hierarchy design works exactly this way. By qualitative analysis the user can detects clusters, correlation and implication among facet values.

6.1.5 Regions of the Ontological Space of Data

The division of the Dimension of Scale into the segments *global*, *cluster* and *local* and the division of the Dimension of Abstraction into *model*, *realization* and *instances* yields nine regions within the OSD, as shown in Figure 6.2. A clean border between the regions can hardly be defined, rather they represents major aspects of the data set.

The names of the nine regions are combinations of the corresponding segments in the dimensions whereby the segment of the Dimension of Scale is put first. In addition, the regions are numbered as indicated in Figure 6.2.

I. Global-Model This region includes data components related to the whole model.

In particular, that is a representation of the defined structures and relationships allowed by the model, like class hierarchies, class relations or the set of possible facet values.

II. Cluster-Model Sub sets of the model are considered in the second region. A sub set is an arbitrary part of the model.

III. Local-Model This region focuses on the observation of a single model component and its immediate context as well as annotations and descriptions on the component. In a faceted classification, the facet values are the most specific entities of the model.

IV. Global-Realization A global analysis comprises the whole model and the whole set of instances. It enables quantitative and qualitative information based on all instances in respect to the complete data model.

The region creates the complete concept lattice of a formal context. It puts every object and every model component in the general relation of a concept lattice. Implications and correlation emerging from this lattice are true in the whole data set.

V. Cluster-Realization This region probably is the most important and used one. Analyzing the whole data set at once can be desired to figure out major trends but requires more resources. In most cases, a particular aspect or subset of instances is investigated. The region combines analytical aspects applied to a subset of instances, facets or values

A sub set of instances or model components yields a reduced concept lattice or even a single formal concept. The lattice shows only a part of the data and hence its implications are only true for a sub set of the data. However, different reduced lattices can focus on different or even overlapping aspects. They can be compared in parallel and extended towards more a differentiated sub set of data.

VI. Local-Realization Each model component is related to the instances it describes. As explained above, this relation has a quantitative representation as the number of instances is important. On the other hand an instance corresponds to multiple parts of the model. For instance, the description of a person, according to her affiliation, residence, age and so forth.

VII. Global-Instances Regarding all instances at once, gives an idea of their extend and possible points of interest. One can detect clusters, topological structures like spanning trees or cycles in a network, composite and hierarchy structures, or general metrics about all instances. However, information involves few knowledge from the model, if at all.

Regarding a visualization, the focus in on showing the whole data set and relations. Topological and composite structures as well as metrics help in creating a purposeful

picture.

VIII. Cluster-Instances Parts of the instances can be structural clusters or arbitrary subsets. Also a filtering according to the model is possible, which happens in faceted browsing.

IX. Local-Instances A single instance may have several literal properties like a name and a URL. They actually correspond to the model since their existence is defined there, but their content is object specific and unique.

6.1.6 Exemplification of the Ontological Space of Data

In order to exemplify and substantiate the Ontological Space of Data, a short classification of two projects presented in Section 3 is given; Relation Browser++ and D-SIFT. The projects are demonstrated in the following analytical way:

1. define and classify data components,
2. define explicit structures among the components,
3. classify visualizations, and
4. describe navigation and transition.

Classifying Relation Browser++

Relation Browser++ is a typical faceted browser. Data components are facets, values and objects. There is no hierarchy among facets or values, nor explicit relations among instances. The classification of this data components is the same as in Figure 6.1.

Facets are selected from a drop-down menu and only three of them can be seen simultaneously. Thus the facet list is settled in the Region Cluster-Model. Relation Browser++ does not deliver a holistic overview on all facets and values at once. Values are also shown as a list, but every value uses a numerical as well as a graphical representation for information from a quantitative analysis.

Results of a faceted search are represented in a list that shows the object name and assigned facet values. The result list can comprise all objects from the data base or just a single one, depending on the current query.

The only view transition that exist in the interface is the change of visible facets.

As seen from Figure 6.3, no explicit information is provided about a realization of the model or correlation among values. To obtain this information the user must browse the data set and compare the values by her own.

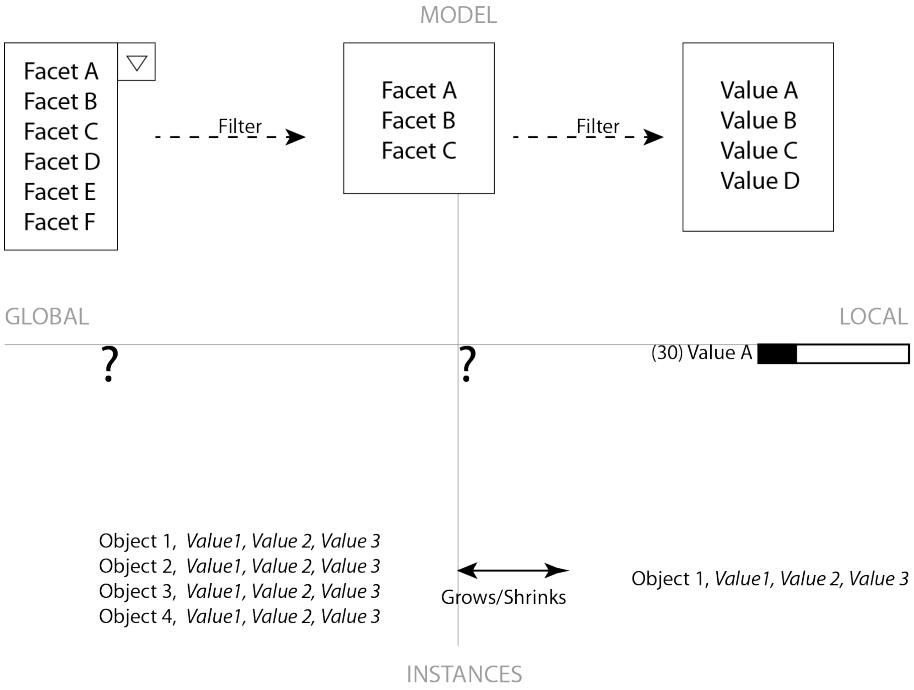


Figure 6.3: Classification of interface components and visualization of Relation Browser++ into the Ontological Space of Data. The black arrow shows a change of the same view, dashed arrows show an influence of one view to another.

Classifying D-SIFT

The four interface components of D-SIFT are (1) drop down menu for selecting the data source, (2) two lists for choosing facets and attributes respectively, (3) a lattice and (4) a list for the currently selected attributes (cf. Section 3.3).

Data components are the same as for faceted browsing: objects, facets and values. There are no explicit relations among objects and no facet hierarchy exists.

Figure 6.4 shows the visualizations and interface components of D-SIFT. Facets and values are shown in two independent lists. These lists can be scrolled so that all facets and all values can be observed. If the user selects values, the concept lattice is adapted and can grow larger or shrink, depending on the values. The lattice belongs to the realization because it shows how the selected values do occur and correlate within the data set. The smallest sub set of instances is represented by a formal concept.

There is no common representation or detailed descriptions of the objects, shown by the question marks in Figure 6.4.

The exemplification of the Ontological Space of Data by Relation Browser++ and D-SIFT shows a method for classifying arbitrary visualization environments in the OSD and evaluates methods for visualizing different data components. It also demonstrated that none of the presented projects cover all regions of the OSD. Even more interesting is that each of them covers something the others do not and that similar visualization patterns occur.

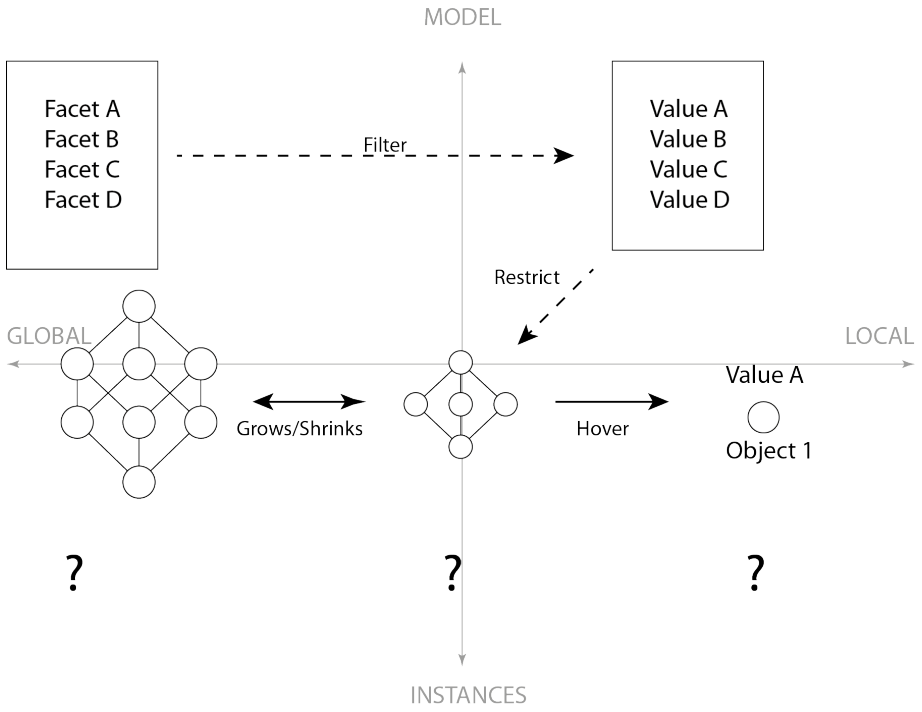


Figure 6.4: Classification of D-SIFT into the Ontological Space of Data.

6.2 Facettice Environment

This chapter presents the general concept of Facettice by means of the Ontological Space of Data. Then, an evaluation of an interface metaphor is made, in order to define a narrative and graphical framework. This section also explains the minor interface compo f Facettice, a Dynamic Taxonomy view and the representation of search results.

Two conclusions result from the definition of the Ontological Space of Data. First, to give a holistic access to a data set, every region in the model has to be taken into account. The second conclusion is that a visualization can cover aspects from different regions, as seen by the growing and shrinking lattice in D-SIFT. Facettice proposes interface components and visualizations for all regions within the model, by keeping the transitions as seamless as possible. Of particular interest is the connection between the model and realization as well as visualization of the realization regions.

It follows a classification of component from the DelViz Visualization Taxonomy into the Ontological Space of Data, considering an integration of faceted navigation and FCA.

6.2.1 Data Components

As explained in Section 5.1, the DelViz Visualization Taxonomy consists of a two-level facet hierarchy and no value hierarchy. Objects are visualization projects with

a title, pictures, a textual description and an URL pointing to the web page.

6.2.2 Relationships

There are no explicit relations between visualization projects in the DelViz data set. The structure of the data model is a strict hierarchy without multiple inheritance. It means that facets have only one super facet and values belong to only one facet.

An implicit relationship is that objects can be classified by arbitrary facet values, also from the same facet. Another relation is that objects are clustered by similarity when the user constrains facets. According to a faceted navigation and browsing process, the query alters. It changes the current instance cluster as described by the navigation modes in Section 4.5. While "traveling" from concept to concept, the user discovers more of the data set.

6.2.3 Views

The data set of the DelViz Visualization Taxonomy divides into three major parts which are the facet and value hierarchy data model, the project instances and the search space made by concepts and represented by a concept lattice. Figure 6.5 gives an overview of the views used in Facettice to cover the whole Ontological Space of Data.

Facet and Value Hierarchy View

The facet and value hierarchy is an important part of every faceted browser and has to be clear and accessible everywhere in the application. In Facettice, a horizontal tree representation is chosen, shown in the upper left corner of Figure 6.5. The hierarchy is comprised of three top facets, than sub facets and finally the facet values. Values can be restricted directly by clicking them.

The tree representation gives a holistic overview of all possible facets and facet values. Expanding and collapsing tree nodes ensure clear overview and detail at the same time. Values are distinguished from facets by an italic font face.

In order to support query preview and allow a quantitative analysis of the model, numbers of remaining objects for each facet value are indicated. This design pattern is well known and heavily used in other applications (cf. 4.6). Numbers at facets indicate the amount of facet values, independent from their level in the hierarchy. Although the different application of numbers in the case of facets and facet values seems confusing, the effect is lowered due to differences in the type face of facets and facet values. All numbers are updated each time the user changes the query, representing the dynamic taxonomy. The whole hierarchy visualization is extensible to an arbitrary amount of facets, values and hierarchy levels.

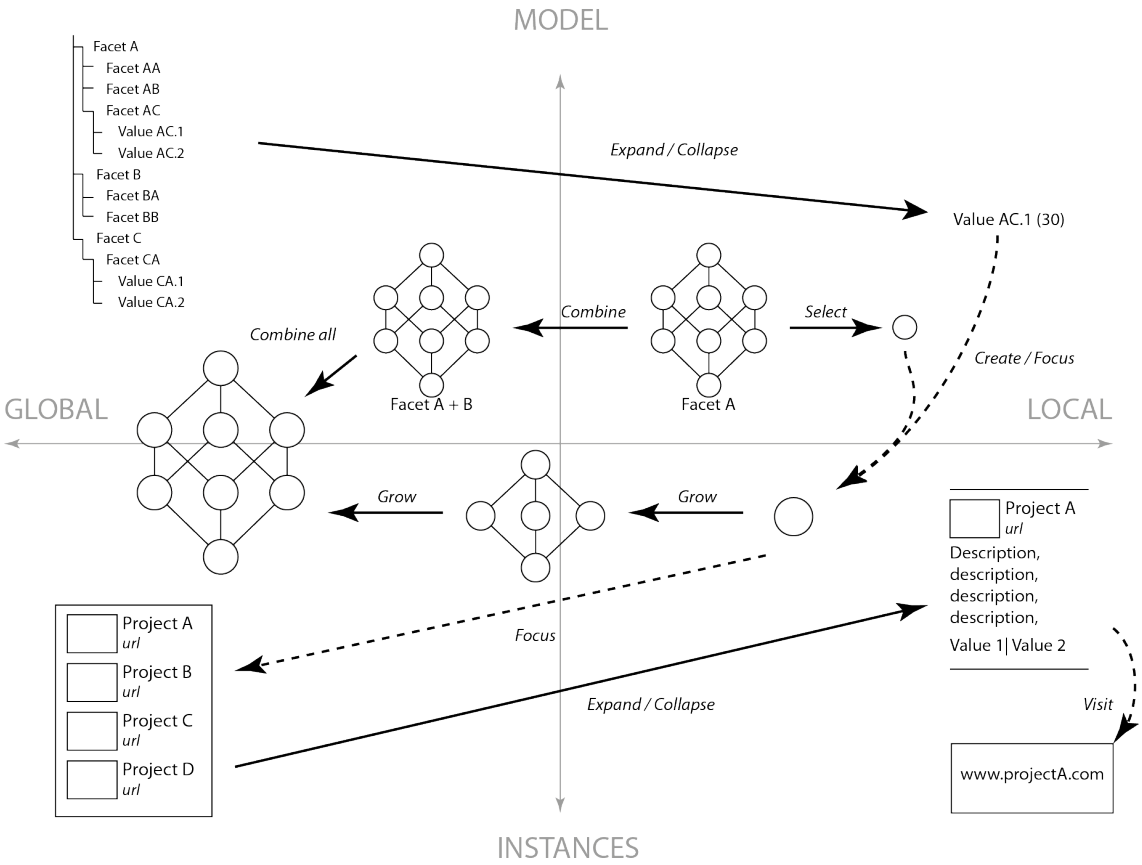


Figure 6.5: Classification of visualizations and interface components of Facettice in the Ontological Space of Data. Black arrows indicate direct transitions while dashed ones describe an influence.

Values of facets which do no longer occur in the current result set, are faded to gray. They are not pruned to keep the general overview of all data model components.

The facet and value hierarchy visualization is a generic solution for delivering a holistic overview on the data model as well as specifying single values and get feedback about further queries in the form of a qualitative analysis.

Result Presentation View

Aspects of regions VII to IX are treated within an own interface component because displaying objects within the hierarchy or a concept lattice would over-stress those visualizations. Dealing with objects in a separate interface component gives larger freedom to represent them in an appropriate manner.

Since there are no explicit relations between the visualization projects they are displayed in a list, ordered alphabetically, for instance. Any project in this list consists of its title and a thumbnail, if available. If not, a "no Image" picture is displayed.

A detailed view, which is required to cover region IX Local-Instances, includes images for projects, a detailed description, applied tags and the URL for accessing the webpage. The webpage can be accessed directly from Facettice delivering additional information. In Figure 6.5 the webpage is shown in the upper right corner.

Big Smart Lattice View

The realization of the complete model can be visualized by the concept lattice of the complete formal context. Figure 6.5 shows this lattice in Region IV Global-Realization.

One way to reduce the complexity of this lattice is to show only those parts that are of interest to the user. While browsing the data set, that is concept by concept, the concept lattice can support him by showing related concepts and create a general context for his search (cf Section 4.7).

The are many to ways to create the concept neighborhood for a given concecpt. While Ulysses uses a fisheye technique, Facettice considers the browser history of the user and consequently builds the complete lattice.

Facet Lattices View

As mentioned in Chapter 4, a formal context can be reduced by grouping attributes. Using facets it is easy to create a lattice according to a single facet. Facettice creates one lattice per facet to enable a focused investigation of the realization of facets in the data set.

Figure 6.5 shows those facet lattices in Region V Cluster-Realization at the border to Region II Cluster-Model. They are, indeed, closer to the model than a Big Smart

Lattice because each lattice represents exactly one facet. The Big Smart Lattice, in contrary is built upon the direct neighborhood of the current result set.

Complex lattices can also be formed when creating a concept lattice from multiple facets.

6.2.4 Transitions and Navigation

Big Smart lattice and facet lattices are very similar and since the Ontological Space of Data defines seamless transitions on the dimensions, a transition between the big lattice in Region IV and the lattices in Region V are imaginable. Building a lattice upon all facets yields the big lattice in Region IV and exploring the whole data space, also yields the big lattice. These transitions can be used in an interface, if desired. However, Facettice keeps Big Smart Lattice and Facet Lattices seperated in the interface. First is for supporting the user in search and exploration while the purpose of the latter is to enable analysis of single facets.

To specify a query, the user can either select values from the omnipresent facet and value hierarchy or he can use the facet lattices. Both results in an adaptation of the Big Smart lattice. In order to make the user independent from the facet and value hierarchy and the facet lattices, he also must be able to restrict facets within the Big Fat Lattice. The Big Fat Lattice as well as the Facet Lattices are described in Section 6.3 and 6.4.

The separate result view is adapted to each new query, as well. By selecting a project, the detailed description is shown and the webpage can be accessed.

6.2.5 The Metro Map Metaphor

Metaphors help understanding complex and unknown ideas or systems by relating them to something known. Since concept lattices are naturally unknown to most people, it is made the attempt to apply an interface metaphor and reuse solutions for equal problems . Beyond increasing usability of concept lattices, the metaphor also serves as a repository for graphical solution in information visualization and further extensions of Facettice.

The landscape metaphor for FCA, was already described by Wille, although in a rather loose way [Wille, 1999]. An extensive application has not been investigated so far. Also the landscape metaphor is ample and has to be exploited carefully.

Since Harry Beck invented "modern" metro map design, those maps are graphically simple by remaining powerful in their function (cf. [Garland, 1994]). A metro map is an (1) overview plan of a territory that (2) can be traveled/cruised in (3) many different fashions, (4) leading successively from one point to another, (5) allows changing routes and contains (6) different districts. (8) Close stations lead to close neighborhoods and sometimes such a map shows (9) landmarks for important places.

The enormous graphical richness and graphical perfection of modern metro maps

Figure 6.6: Detail of the current parisian metro plan showing graphical codification of stations and lines.

represent a huge graphical repository reminding Mackinlay's composition algebra (cf. [Mackinley, 1986]). It can be used to enhance conventional lattice representation in the way of information codification as well as visual attraction. Graphically a metro maps need to distinguish between different types of stations (differing in form and size), track lines (differing in color, thickness and texture) and terrain (river, landmarks).

Most of these aspects also hold for concept lattice representation and graphical solutions can be adapted. Figure 6.7 shows a simple graphical mapping by using concepts as stations and mapping attributes to metro lines. A metro attribute line is defined by connecting all concepts where the particular attribute is part of the concept intent. Indeed, the result reminds more a network than a linear metro line but also metro lines contain furcation and cycles. This fact may be the most confusing one when applying the metro map metaphor to concept lattices. To lower visual clutter, top and bottom concept have been omitted since in this example they bear no additional information.

The intent of a concept can easily be determined by following each related metro line to the top. Another effect is, that the distribution of attributes within the lattices is visible. In the example on can see that `alcoholic` and `nonAlcoholic` do not correlate and that `hot` implies `nonAlcoholic`.

Districts within a metro map can refer to attributes, since the attribute distribution defines the layout, or can emphasize concept clusters. Clusters in lattices are a closely related concepts. In Figure 6.7, three districts are shown, emphasizing clusters that are specified by particular attributes and at the same time giving the lattice a very particular appearance. s

The metro map is not an unambiguous metaphor for lattice visualization. A metro map serves for estimating the shortest path from one station to another. Although in lattice visualization, such a question is of interest concerning the similarity of two concepts, the continuos branching can be misleading. Another problem is the amount of attributes and the fact that the intent of infimum close concepts, is large. This results in a huge correlation of colors and reduces readability significantly.

The main motivation for applying the metro metaphor to lattices is not to copy

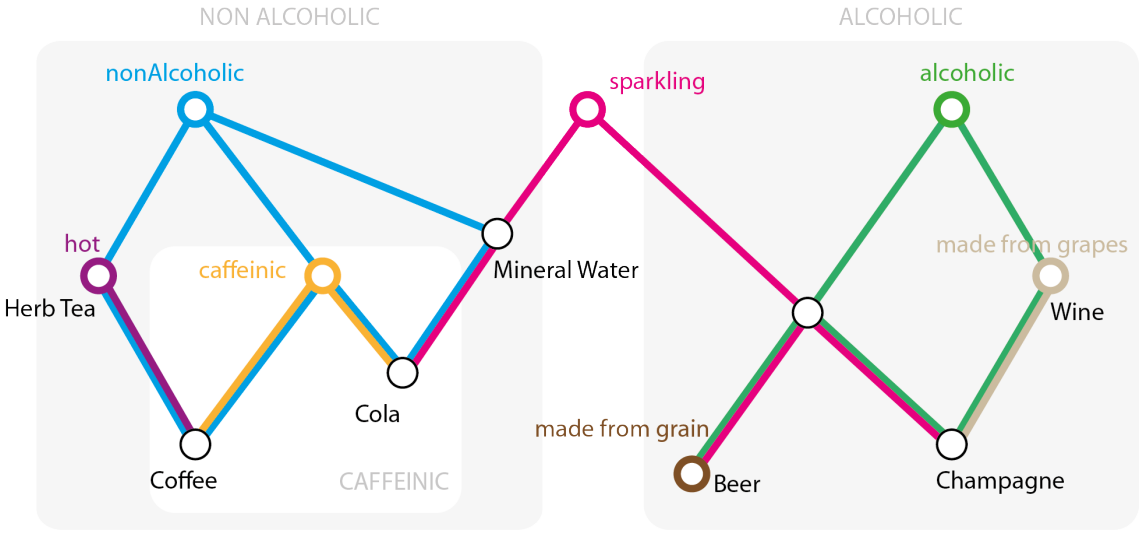


Figure 6.7: Application of the metro map metaphor to concept lattices. Each attribute is represented by a colored line that connects all concepts where the particular attribute is part of the intent. The gray and white squares in the background emphasize clusters.

metro maps and create one big colored lattice. The idea is to use colored lines, stations and districts in order to approach a better understanding and information codification in concept lattices. The following sections refers back to this metaphor and address particular problems.

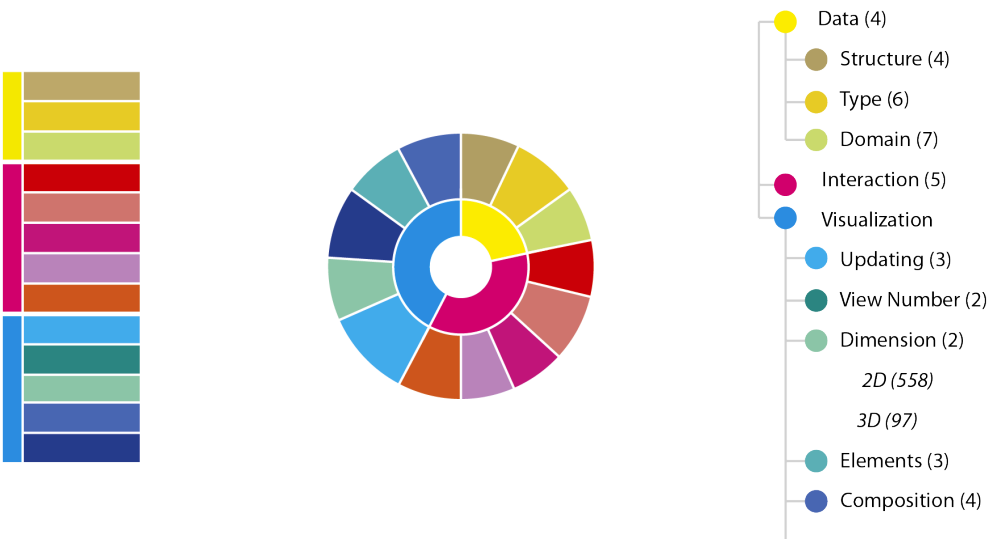
Application of the Metro Map Metaphor

Color can not be applied to facet values because the DelViz Visualization Taxonomy contains 63 facet values. It is impossible to assign a single color to each of them that remains distinguishable.

Facets can be color-coded, which is a common design patterns in faceted browsers (cf. Section 4.6). Colors serve for two purposes. They enables an application wide consistent identification of facets as well as faster recognition. In terms of Bertin's and Mackinlay's classification of data, facets are *equidistant* or *nominal* (cf. [Bertin, 1973, p.43ff] , cf. [Mackinley, 1986, p. 111]).

Colors are equidistant as well and do not inherit a natural order. Other visual variables either imply an ordering, such as *lightness* or *size* or are sometimes difficult to apply like *position*. Although color is limited in its extend to represent equidistant data, an advantage is that color can represent hierarchical data to a certain extent. A natural order among colors exist only according to the wave length of light. This order is sometimes used in information visualization to show large ranges.

For the Visualization Taxonomy the application of colors works well as demonstrated in Figure 6.8. The figure shows three different realizations of a color hierarchy. The right representation is realized in Facettice. Values are indicated by the color of its direct facet and are shown without color in the hierarchy.



(a) Horizontal Hierarchy (b) Radial Hierarchy (c) Tree in Facettice

Figure 6.8: Color hierarchy for facets as used in Facettice. The left figure illustrates a horizontal layout, the center figure illustrates the appearance using a radial layout and the right figure represents the integration of colors into the facet and value hierarchy tree of the interface.

6.2.6 General Layout

Figure 6.9 shows the layout of Facettice with its components (A) facet lattices showing one lattice per facet, (B) the facet and value hierarchy, (C) the Big Smart Lattice and (D) the result list showing the resulting visualization projects. The facet lattices and the context lattice are the two main parts of *Facettice* and are described in detail in the next chapters.

The four components are furthermore synchronized by *brushing and linking* (cf. [Keim, 2002]). It means that a selection in one view causes a highlighting in the other view. Also, if one view changes, the others are changed as well. This allows a consistent view on the data and enables focusing on particular views. Facet lattices, facet and attribute hierarchy as well as the result list can be hidden by the user, in order to better use the screen real estate for the big lattice.

6.3 Facet Lattices

The Facet Lattice component contains one lattice for each facet that has no further sub facets. The attributes that define the formal context, are the facet values of this particular facet. This component arises out of the idea to show the interrelations of facet values. A hierarchy can not show correlation among its values. This is done by this visualization and has many benefits and leads to new approaches in visualization and interaction.

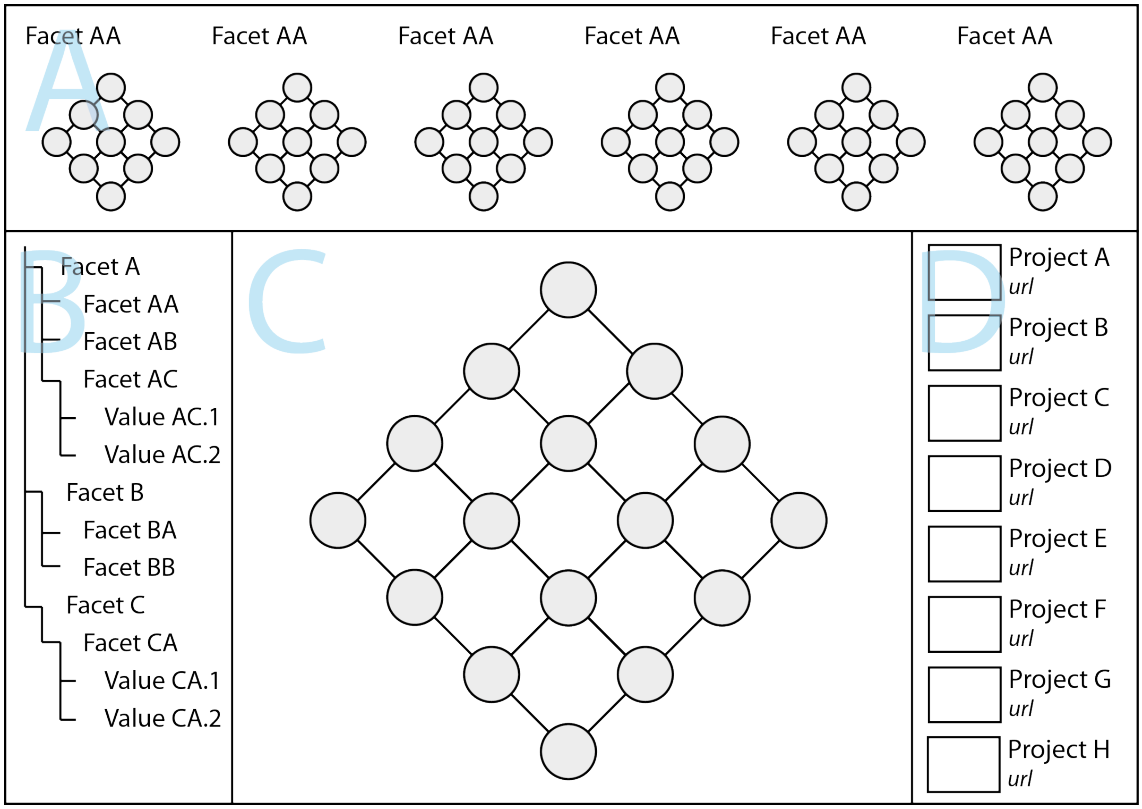


Figure 6.9: Principle layout of the Facettice interface containing the components (A) Facet Lattices, (B) facet and value hierarchy, (C) Big Smart Lattice and (D) result presentation.

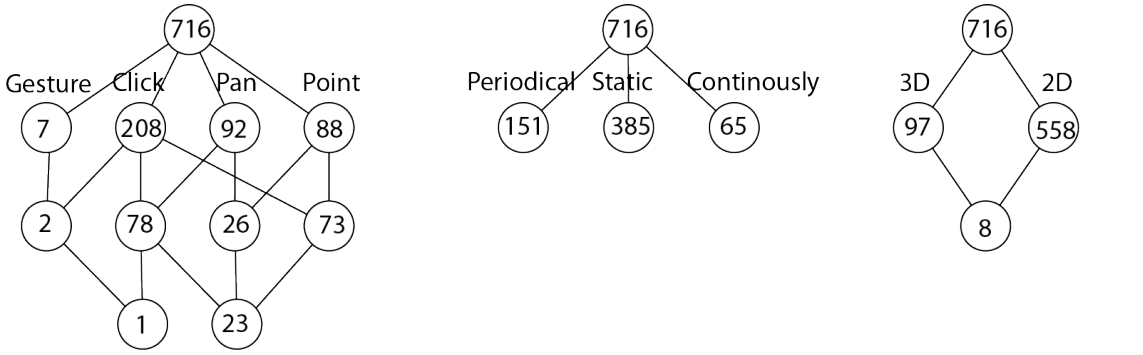
In FaIR, Priss already uses lattice structures in order to display and retrieve objects from a faceted data set (cf. 3.3). But while she relies on a taxonomy, Facettice shows complete lattices. Figure 6.10 shows three concept lattices created from the facets `Interaction/Control Move`, `Visualization/Updating` and `Visualization/Dimensions` from the DelViz Visualization Taxonomy. The top concept contains all objects of the data set and the numbers within the concepts represent their extent. The bottom concept is pruned if it is not realized.

A comparison reveals different structures and characteristics that occur among facets and let draw conclusions about their semantics in terms of a realization as described with the Ontological Space of Data. The facet `Updating`, for instance, is mutually exclusive. There is no interrelation between the facet values and the lattice looks like a taxonomy. In contrast, `Control Moves` contains well distributed facet values. A value from facet `Dimensions` is assigned to almost all projects and they are either 2D or 3D[1]. Just in eight cases, both dimensions appear.

Besides the amount of formal attributes, which is given, the shown lattices and hence facets, differ in the following individual aspects:

1. **Amount of concepts** Apart from the attribute amount itself, many concepts indicate high interrelation. Although facet A in Figure 6.10 has only one attribute more than facet B, its lattice has more than three times more concepts,

[1] 97 + 558 - 2*8 = 639 objects are either labeled 3D or 3D



(A) Interaction/Control Move

(B) Visualization/Updating

(C) Visualization/Dimensions

Figure 6.10: Schematic view on three facet lattices from the DelViz Visualization Taxonomy

neglecting the top concept. There is a higher rate in combining values of this characteristic. For the particular facet or characteristic, it means that Control Moves are independent from each other, while every visualization can only be updated according to one method.

2. **Depth of Lattice** The depth of a lattice in dependence of the amount of attributes, is an indicator for the completeness of combination. Lattice A shows that the data base does not contain any objects using all interaction types. Lattice C is complete and all possible combinations of values occur in reality.

3. **Interrelatedness** The amount of concepts is not necessarily the only indicator for relatedness of a facet. Lattice B and C have the same amount of concepts, nevertheless lattice B shows no relatedness among the facet values, while the facet `Dimensions` share at least eight common objects.

4. **Concept size** In Facettice, the top concept always shows the number of all objects within the data base in order to be consistent among all Fact Lattices. Thus, it also contains objects that do not satisfy one single value from this facet. In general, the size of the extent of concepts is very important and helps distinguishing general and specific facet values. In lattice C, there are more than five times more objects within the data base that have only two dimensions than three.

5. **Balance and Clusters** None of the example lattices is *balanced*, except a sub lattice from lattice A built by considering only `Click`, `Pan` and `Point`. A facet is called *balaced* if all its values are applied in equal number. This sub facet can also be denoted as cluster, because its inner interrelation is far stronger than its outer one. Clusters indicate strong correlation among its items.

   An equilibrated facet describes a characteristic which's values are equally distributed and occur in arbitrary combinations like combinations in playing dice.

6. **Correlation** Lattice A shows a correlation between `Pan` and `Point` in approximately one fourth of all cases. This is indicated by the concept containing 26

objects.

7. **Implication** In Lattice A there is an almost complete implication between the the values `Point` and `Click`, because 82% of all projects that use pointing, also use clicking. This is indicated by the concept containing 73 objects.

Most of the characteristics are hard to figure out when lattices are presented as in Figure 6.10. For that reason, more information must be encoded into the lattice representation. The next section describes which information is used and how it is encoded. Interaction techniques with facet lattices are described at the end of this section.

6.3.1 Information Codification

In order to answer the above listed questions, the following list shows all data and meta-data that is available to be encoded visually. As already mentioned, representations of formal objects are treated apart from the lattice and only their amount is considered here.

- **Relative concept size (support)** The support is the relative size of the concept extent in comparison to all formal concepts in the database. The support is indicated as percentage value (cf. Section 2.3.2).
- **Absolute concept size** This measurement helps clarifying what part of a concept is actually apparent. Regarding Lattice C in Figure 6.10 projects have been labeled `2D` of which eight are also labeled `3D`. Thus, 89 projects are solely labeled `2D`. While in this particular lattice, such a calculation is easy, this calculation for lattice A is almost impossible by hand.
- **Intent** Figure 6.10 only shows attributes at the corresponding attribute concept. In large lattices it is hard to follow their distribution among all concepts.
- **Size of intent** Noticing the amount of attributes at a glance helps estimating correlation and implication.
- **Confidence** Confidence is the percentage of common objects that in relation to the extend of the parent concept. It is a sign of implication, as in the case of `Point` and `Click` in lattice A.
- **Common objects** The absolute number of common objects between parent and child concept helps determining flows and clusters within the lattice.

The graphical components of a lattice are the same like in ordinary networks; Nodes, relations and labels. Labels are embodied directly into nodes or relations. In the following an information codification in nodes and relations is described.

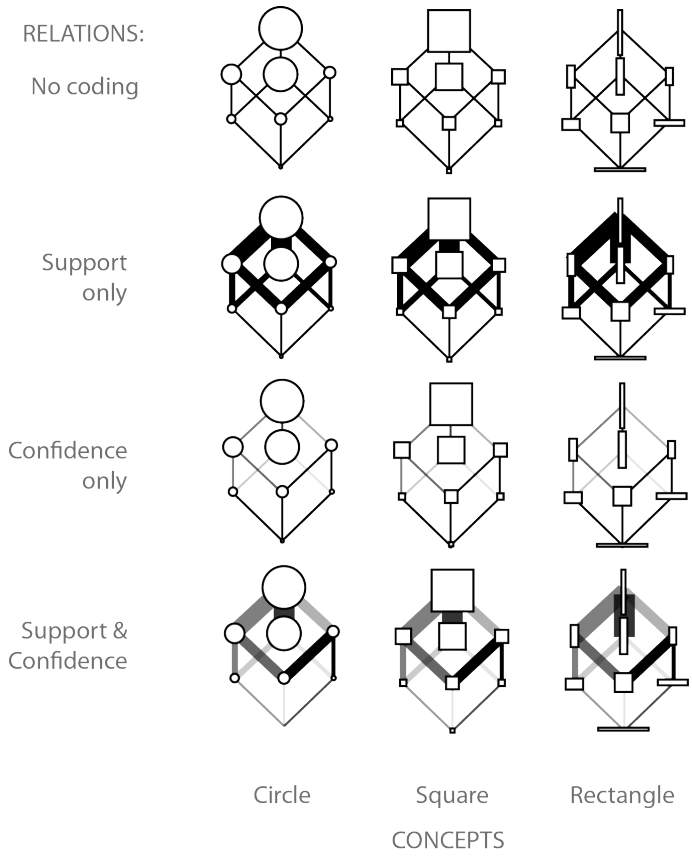


Figure 6.11: Comparing possibilities for concept and relation representation.

Concepts

Most lattice representations use circles for representing concepts. Only Ulysses and Conflexplore use rectangles for better placing object and attribute names inside (cf. Chapter 3). It has been made a comparison between three shapes, *circle, square* and *Rectangle*. In order to compare which one shows best the differences in size. The size of concepts depends on the size of their extent, as explained later.

Circle and square are trivial shapes while the rectangle results from mapping the size of objects to the vertical extension of the rectangle and the size of attributes to the horizontal extension. Consequently, the resulting lattice shows a clear transition of vertically oriented rectangles on the top of the lattice, over almost squares to horizontally oriented rectangles at the bottom. Although this mapping delivers promising figures, it is hard to compare sizes of extents. The perception is biased by the area of the rectangles. The concept directly above the bottom concept seems larger then the concept above but has fewer objects. Further investigation on the rectangle shape must be done but is beyond the possibilities of this paper.

For *Facettice* the circle was chosen, since it is closer to the metro metaphor and makes the lattice appearance calm, in comparison to square and rectangle.

According to the investigations of Bertin and Mackinlay *quantitative* data is best coded visually by *size* or *length* (cf. [Bertin, 1973]). Several mapping functions can

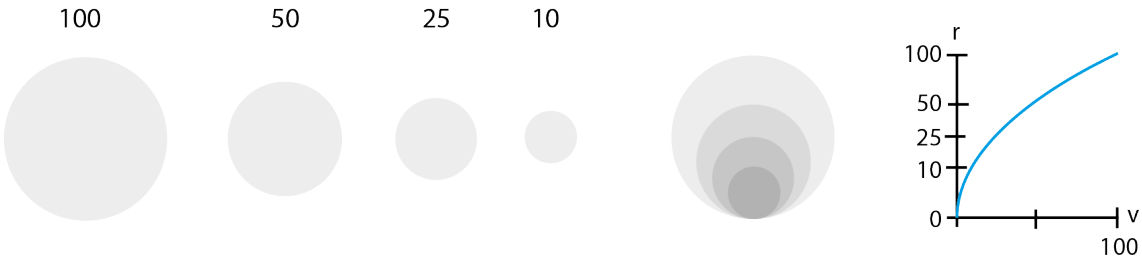


Figure 6.12: Demonstration of mapping a value v to the radius r of a circle by using the function $r = \sqrt{v}$. The nested circles show the difference in size in an alternative view and the diagram shows the progress of the mapping function.

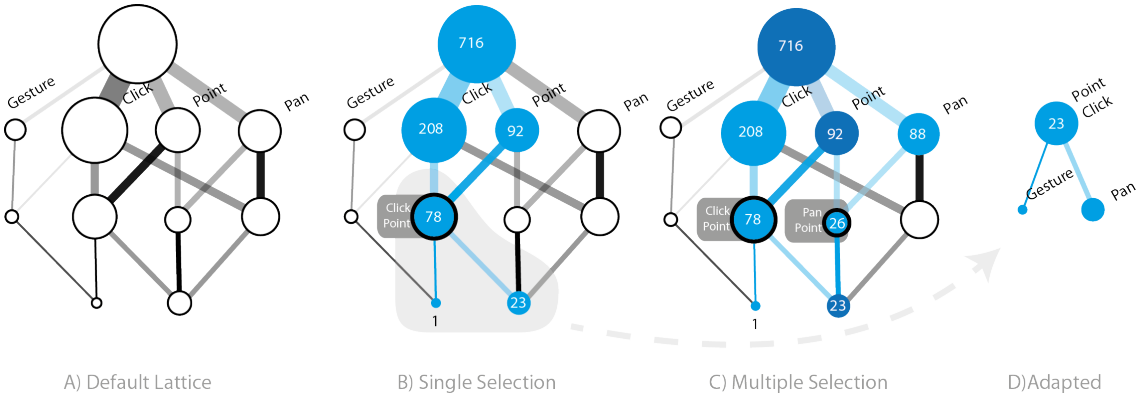


Figure 6.13: Four states of a visualized Facet Lattice; default, selection of one concept, selection of two concepts and adaption.

be applied to map the number of objects v to the size A of a circle. The formula for the circle surface is $A = 2r^2\pi$, which becomes $r = \sqrt{\frac{A}{2\pi}}$ and conforms to the ratio $r = \sqrt{v}$. Figure 6.12 demonstrates the difference of circle size using this function. This mapping corresponds well to human perception while maintaining low calculation time. It is applied for the nodes in the Facet lattices as well as for the nodes of the Big Smart Lattice, which is described in Section 6.4.

Relations

Relations can encode confidence and the amount of common objects. Since confidence is a relative value, it can be mapped to *lightness*. On a white background, a black relation signifies $conf(c) = 1$ and white means $conf(c) = 0$. The number of common objects corresponds directly to the support of the child concept and is best mapped to the thickness of relations. The last three rows in Figure 6.11 demonstrate these mappings. The last row shows the combination of support mapped to thickness and confidence mapped to lightness. The last row also shows, that in combination with confidence and support, circles are best to represent concepts. The left most lattice in Figure 6.13 shows the visualized lattice from facet `Interaction/Control Moves`.

6.3.2 Interaction with Facet Lattices

The primary interaction purpose of facet lattices is the selection of concepts. By selecting an arbitrary concept the corresponding intent is added to the current query of faceted values and the resulting objects are shown within the result presentation component. Figure 6.13 shows the four states of a lattice *Default*, *Single Selection*, *Multiple Selection* and *Adapted*. An additional interaction technique is to combine facet lattices.

Highlighting Every concept can be hovered which causes all parent and all child concepts to be highlighted as seen in Figure 6.13 Lattice B. A highlighting of the parent concepts shows where the position of the particular concept in the attribute hierarchy. This is important since attributes are only shown once in a lattice. Child concepts indicate the distribution of other attributes among the extend of the hovered concept.

Selection Selecting a concept leads to a fixation of the highlighted concepts. This allows to highlight other concepts and see concepts which are highlighted by both selections, as shown by Lattice C. Another effect of multiple selection is that queries can be visualized, as explained in Section 6.3.3.

Adaptation As soon as the current query changes, either by selecting a concept in a facet lattice or in any other way, all Facet Lattices are adapted to the new sub context. The new sub context contains only objects that satisfy the new query. Lattice D shows the adapted lattice for selecting the concept with the intent {`Click`, `Point`} in Lattice B.

Combination Until now, only one lattices per facet has been considered. This makes the question *"Are red cars faster than blue ones?"* impossible to answer because *color* and *speed* are two different facets. Facet Lattices can be combined and create a new concept lattice that shows the formal context based on both sets of facet values. Values from both facets are distinguished by the color of the corresponding facet.

In *Facettice* this is done by dragging one lattice onto another. A new lattice is created from the sub context containing the attributes from both facets. Facets in the new lattice are distinguished by color as shown in Figure 6.14, where the concepts *ab* and *bd* are selected.

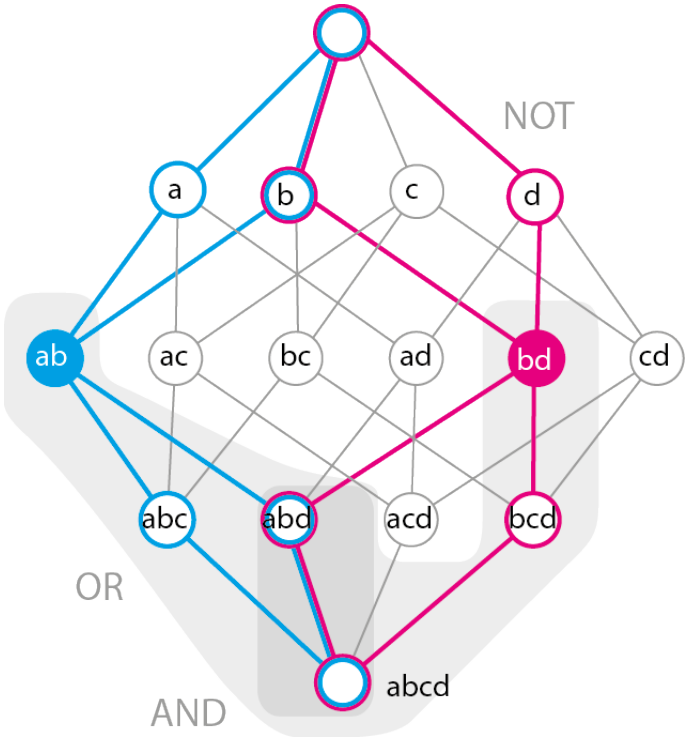


Figure 6.14: Two selected concepts (filled circles) within a Facet Lattice and the three regions AND, OR and NOT which are created. The letters are the attributes in each intent. Colors are do not correspond to facets, they distinguish both concepts.

6.3.3 Query Visualization

When two concepts p_1 and p_2 are selected within a lattice, the lattice divides into several *regions*. A *region* is a group of connected concepts. Two concepts *ab* and *bd* are selected in Figure 6.14 and three particular regions are shown. Those regions correspond to boolean queries of the intent of the both *pivot concepts*. In Figure 6.14, the concepts labeled according the attributes in their intent.

OR-Region The OR-Region contains all child concepts of the pivot concepts as shown in Figure 6.10. Formally the OR-Region is described by the query $int(p_1) \vee int(p_2)$, whereby $int(c)$ is the boolean AND query of all attributes from the intent of concept c. If an intent of a lattice concept matches this query, the concept is part of the OR-Region.

In the example in Figure 6.10 $p_1 = (\{ab\}', ab)$ and $p_2 = (\{bd\}', bd)$. The concepts of the OR-Region are $ab, bd, abc, abd, bc, abcd$ because their intents match at least the query $(a \wedge b) \vee (b \wedge d)$. The concepts *abc* and *abcd* contain the additional attribute c. One purpose of regions is to group concepts and hence attributes. The example shows that the objects in the OR-Region also are classified under c.

The objects of this region are equally retrieved by the conjunction of the extents of the pivot concepts, formally $ext(p_1) \cup ext(p_2)$, where $ext(c)$ is the extent of concept c. Table 6.1 summarizes the formalization for all regions described in this

Region	Boolean Query	Set Operations
OR	$int(p_1) \vee int(p_2)$	$ext(p_1) \cup ext(p_2)$
	$(a \wedge b) \vee (b \wedge d)$	$\{ab\}' \cup \{bd\}'$
AND	$int(p_1) \wedge int(p_2)$	$ext(p_1) \cap ext(p_2)$
	$(a \wedge b) \wedge (b \wedge d)$	$\{ab\}' \cap \{bd\}'$
NOT	$\neg(int(p_1) \vee int(p_2))$	$ext(T) \setminus (ext(p_1) \cup ext(p_2))$
	$\neg((a \wedge b) \vee (b \wedge d))$	$allConcepts \setminus (\{ab\}' \cup \{bd\}')$

Table 6.1: Query regions inside the concept lattice in Figure 6.14 for the concepts $p_1 = (\{ab\}', ab)$ and $p_2 = (\{bd\}', bd)$

section.

AND-Region This region contains all concepts whose intent satisfies at least the query $int(p_1) \wedge int(p_2)$. That are all concepts which are children of both pivot concepts. In Figure 6.14 those are concepts abc and $abcd$.

NOT-Region By definition, the NOT-Region contains all concepts and objects that are not within the OR-Region.

Every concept can be classified into one of these three regions. The extents of the concepts correspond to the query of their region. Visualizing the regions shows the user not only if there is an AND-Region and how the OR and NOT-Region are shaped but also what further attributes occur inside every region. The lattice in Figure 6.10 contains all combinations of attributes, thus the attribute c is contained in every region. But, examples are possible where c does only occur in the AND-Region, which means $a \wedge b \wedge d \Rightarrow c$.

Query visualization is not limited to facet lattices and can applied to all concept lattices. Although the definition of regions is not yet completed, this section demonstrated the idea of lattice regions and query visualization. The three regions can likely be subdivided to yield more precise statements about concepts, attributes and objects within. However, a deeper investigation is beyond the scope of this thesis.

6.4 Big Smart Lattice

The Big Smart Lattice is the second major part of Facettice. As described in Section 6.2.3, the complete lattice for the whole data set results in a "Big Fat Lattice (BFL)", to adapt Schraefel and Karger's notion of the "Big Fat Graph" (cf. [Schraefel and Karger, 2006]).

The Big Smart Lattice (BSL) shows only the part of the BFL that is the context of the current focus. While the user explores the data, the BSL grows and shows visited concepts and its neighbors. That way, an individual concept lattice is created that reflects the user's way and her perspective on the data.

The BSL supports search and exploration because it shows neighbors of concepts and the concepts the user has visited. Data analysis is supported in two ways. First, concepts can be compared directly and second, arbitrary facet values can be combined.

This section discusses the four conceptual parts of the Big Smart Lattice, which are (A) methods for successive construction, (B) the representation of concepts, (C) the representation of relations between concepts and (D) a visualization of the users search history which comprises visited concepts. All representations must be integrated in the same visualization and their development influences each other.

6.4.1 Construction Methods

Iceberg lattices, D-SIFT and Ulysses present different techniques to reduce concept lattices. They show only the part that is currently most interesting to the user (cf. Chapter 3).

Facettice makes a different approach. If the user changes the current query by selecting a concept from the facet lattices or the facet and value hierarchy, a new concept is created and added to the BSL. If the particular concept already exists, it is focused. In both cases the focused concept represents the current query and the user's "position" within the lattice. In addition to the new concept, its neighbors and relations are added implicitly. The process of implicitly adding concepts or relations is called *inference*.

The BSL starts with the top concept which comprises all objects and an empty intent. Is is the common reference for all further concepts and queries. Four ways, or levels, have been developed to construct a BSL and infer neighbored concepts. The simplest way does not infer any neighbors and only adds new concepts which correspond to a user query. The other levels increase the number of inferred concepts and relations.

No Inference

Without inferring, only those concepts are added to the lattice, which arise from a new query. The left four lattices in Figure 6.15 illustrate the lattice creation without inference. Concept as well as relations are created explicitly by the user. This method causes no confusion but neither delivers any additional information about the relation of concepts. Even if concept b in Step 3 is related to the top concept T, the relation is not drawn. Only if the user proceeds to the top concept it occurs.

This construction level only visualizes the way the user has taken. It is a direct

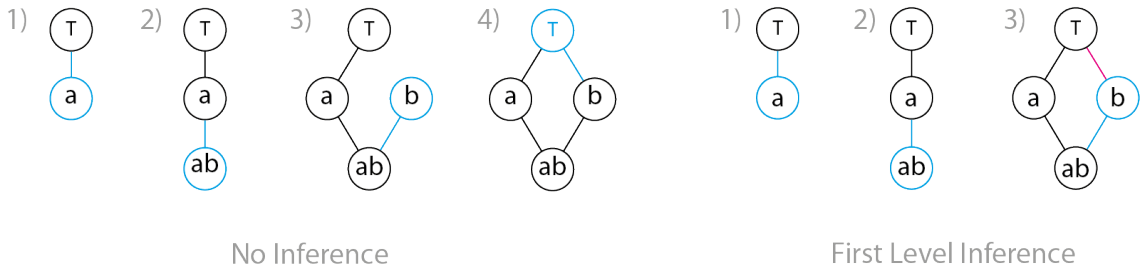


Figure 6.15: Demonstration of the lattice construction using *no inference* (left) and *first level inference* (right). The user first poses the query *a*, then zooms in to *ab*, zooms out to *b* and in the left case zooms out again to the empty query. Blue lines mark explicit relations, the red line is an inferred relation.

representation of the Berry-picking model (cf. Section 2.1.2).

First Level Inference (Relation Inference)

On a first level only direct relations are inferred. Two concepts are directly related if they are in a sub concept relationship. The right three lattices in Figure 6.15 illustrate a *first level inference*. In contrast to *no inference*, the user does not need to zoom out to the top concept T in order to create the relation between concept T and b. The relation is inferred because b is a sub concept of T. Algorithm 1 shows *first level inference* in pseudo code.

Algorithm 1 First Level Inference

```
1: for all concepts do
2:    isRelated := subConcept(newConcept, concept)
      OR superConcept(newConcept, concept)
3:    if isRelated then
4:       createRelation(newConcept, concept);
5:    end if
6: end for
```

A *first level inference* does not create additional concepts, it keeps the lattice small and adds relations that are obvious. It creates a more contextual picture of the lattice than achieved without inference.

Second Level Inference (Attribute Inference)

On a second level, concepts are inferred only if a new attribute occurs in the lattice as shown in Figure 6.16. Algorithm 2 shows the algorithm. As soon as the attribute b is added to the BSL context, a concept with the intent b is created and related to the concept that was created by the user, ab. If the user creates the concept abc, the concept c is added to the lattice in order to relate abc to T.

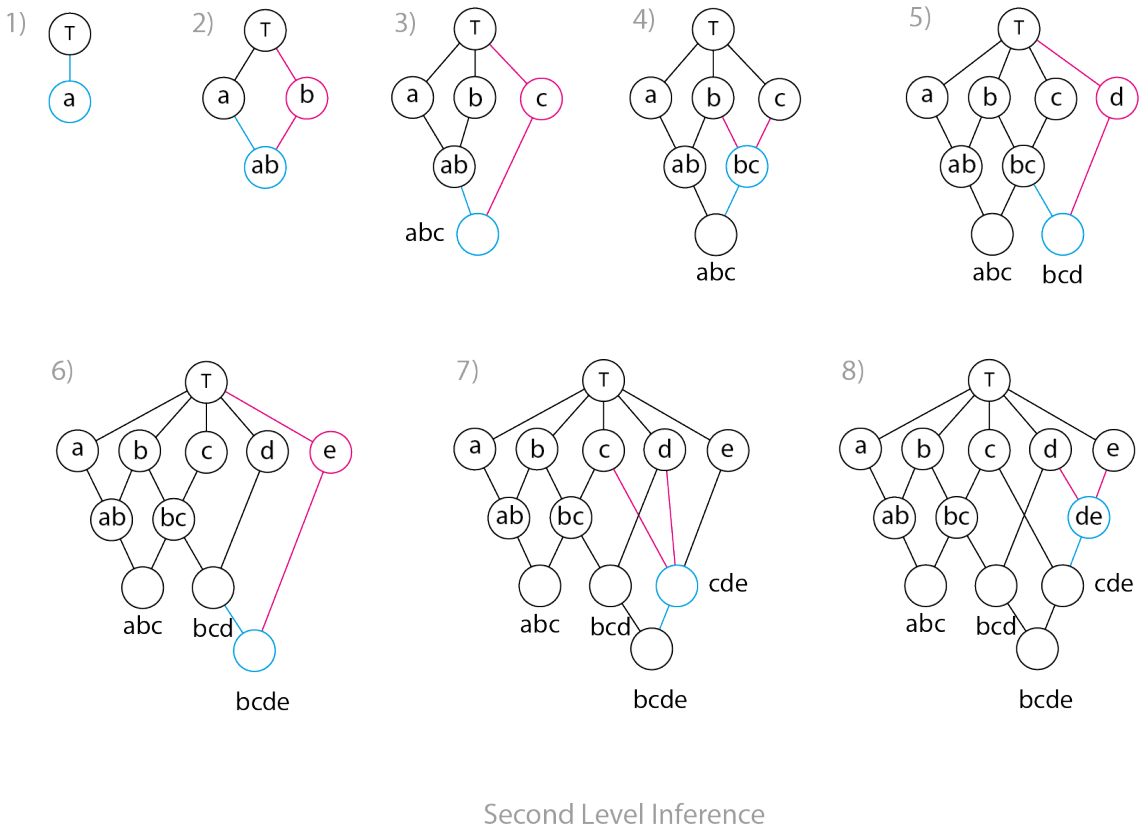


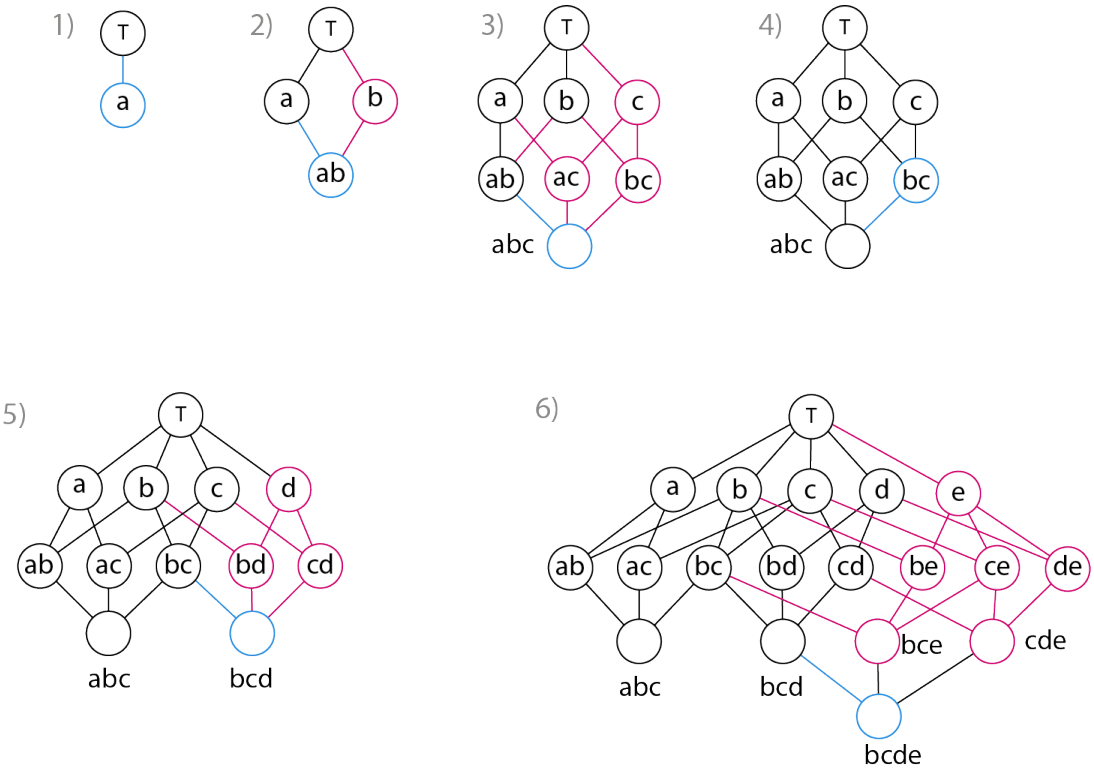
Figure 6.16: Illustration of a second level inference for constructing a Big Smart Lattice

No concept is inferred if the user does not add new attributes to the context as in steps 4, 7 and 8. In that case, relations are inferred based on *first level inference*. As shown in Line 7 of Algorithm 2, after inferring on level one, transitive relations must be pruned.

Shown in Figure 6.16, the resulting lattice stays small because few concepts are inferred. The inferred concepts create a minimal but useful frame for the user. If concepts are inferred he widens his current context and if no concepts are inferred, he remains within the frame spanned by the attributes.

Algorithm 2 First Level Inference

```
1: if newAttributeAppears() then
2:     attributeConcept := createAttributeConcept(newAttribute) ;
3:     createRelation(topConcept, attributeConcept) ;
4:     createRelation(newConcept, attributeConcept) ;
5: else
6:     inferenceFirstLevel();
7:     adaptTransitiveRelations();
8: end if
```



Third Level Inference

Figure 6.17: Illustration of a third level inference for constructing a Big Smart Lattice

Third Level Inference (Neighborhood Inference)

Second level inference causes inconsistency in terms of actual lattice theory. When a relation to an attribute concept is inferred, no concepts in between are ignored. The only rule *second level inference* follows, is that each concept must have super concepts for each attribute from the intent. It is not required to infer *all* super concepts. But, this results in an inconsistent lattice. In Step 3 of Figure 6.16 concepts *ac* and *bc* are missing. If $c \Rightarrow a \wedge b$, there would be no concept c.

To solve this problem, *third level inference* infers all upper concepts, as illustrated in Figure 6.17. Step 3 demonstrates the difference to *second level inference*, because concepts *ac* and *bc* are inferred. Also Step 5 calculates the complete lattice above the new concept. Concepts that are below the new concept are not inferred to not calculate the complete lattice.

The algorithm is simple as described by Algorithm 3. First, a new sub context from the intent of the new concept is created. In this sub context, the new concept is the bottom concept, because its intent contains all attributes from the sub context. Finally, the sub lattice is included into the BSL.

Algorithm 3 First Level Inference

```
1: subContext := createContextFrom(intent(newConcept));
2: subLattice := createLatticeFor(subContext);
3: insert(subLattice);
```

Fourth Level Inference (Complete Inference)

The most complex inference is to calculate the complete lattice every time a new concept is added. If all attributes are used in a query, the "Big Fat Lattice" is obtained. Figure 6.18 demonstrates the lattice after 5 steps.

This method creates the maximal context for each concept. It gives a complete picture of all concepts which exist for the attributes used for querying.

Figure 6.19 directly compares all four inference levels for adding concept *bc* and then *bcd*. The blue concepts and relations are created explicitly, while red concepts and relations are inferred. A direct comparison reveals the advantages and disadvantages of every inference level.

By increasing the inference level, the complexity of the lattice grows faster but delivers a more comprehensive picture of the context. Lower or *no inference* focuses only on the current concept and its context while third and *fourth level inference* focus on completeness and consistency of the lattice. Consequently, the inference levels represent a fine grained transition from supporting focalized search and history visualization (*no inference*), exploratory search (*first* and *second level inference*) until analysis (*third* and *fourth level inference*).

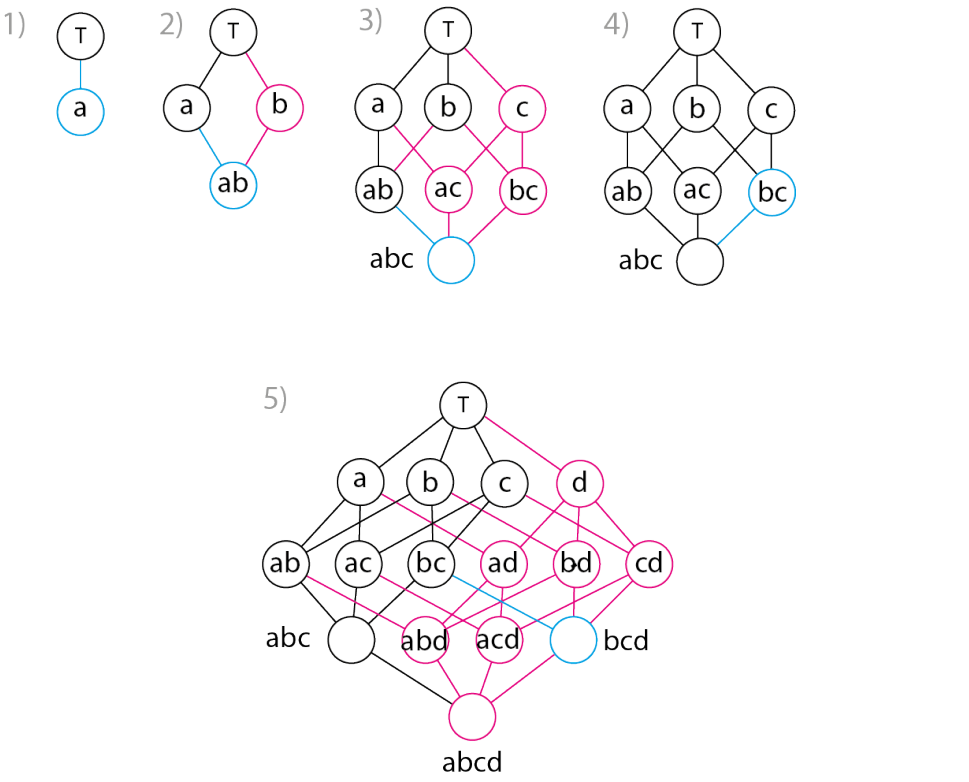


Figure 6.18: Illustration of a fourth level inference for constructing a Big Smart Lattice.

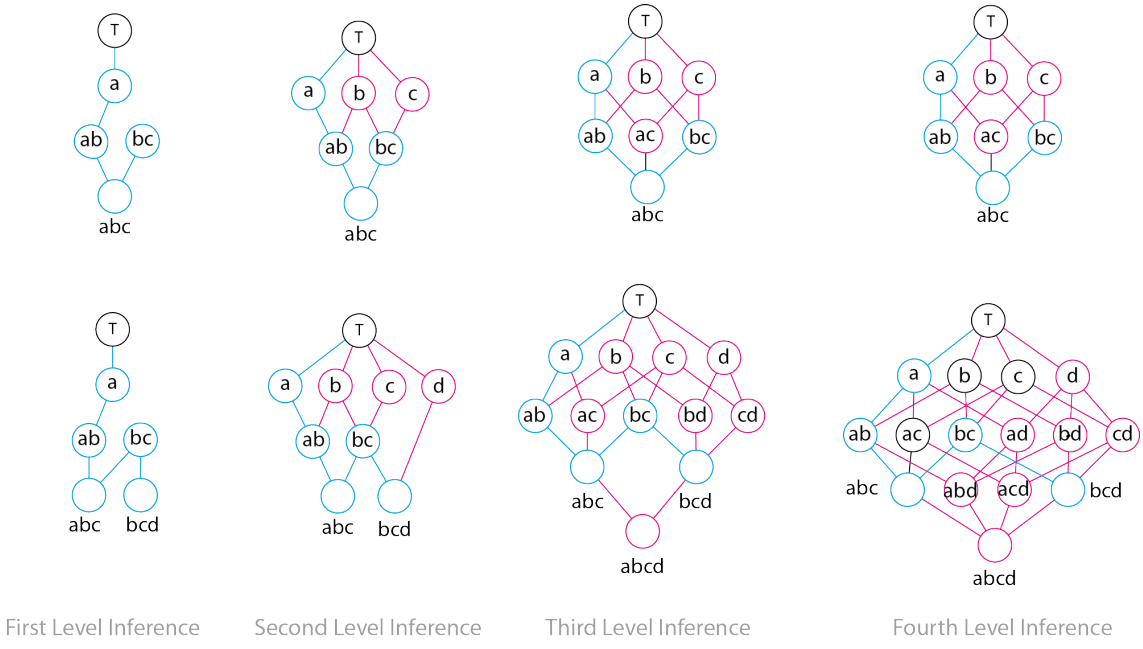


Figure 6.19: Comparing the results of all inference levels in parallel.

6.4.2 Concept Node Representation

Since the BSL shows a resticted part of the BFL, information about missing concepts must be provided. The solution discussed here, is the integration of attributes along with their facets into the concept representation. It gives the user (1) an idea of what the current concept is about, (2) possible paths to related concepts and (3) a better look and feel for interacting with the BSL.

One can make up three classes of attributes for every concept (A) its intent, (B) attributes within the intent that are completely implied by all the others and (C) attributes that are not part of the intent but occur at at least one object from the extent. Depending on the general lattice representation read from top to bottom, in the following, attributes in the intent are called *incoming* and attributes from C are called *outgoing*. This notion works fine for user interaction since he "reaches" a concept by specifying its intent and "leaves" it by adding an attribute which is not yet part of the intent. Attributes under B are part of the intent and are treated equal to the others.

The best way to integrate attributes and their occurrence into a circle concept representation is by laying them out in a circle. Because all attributes are equal, the circle segments are of equal size. The attribute and facet hierarchy is drawn in the style of a radial hierarchy as used in Sunbrst (cf. [Stasko and Zhang, 2000]). Figure 6.22 shows the final representation of the facet and value hierarchy.

Three ideas of encoding information into a concept representation are developed and discussed in Figure 6.20. All are designed in order to best use the space, give a hint on their distribution in the BFL and make visual linkage to other concepts comprehensible (cf. Section 6.4.5). The latter fact is discussed below.

Inner-Outer Representation Shown in Figure 6.20, *incoming attributes* are displayed on an inner circle while outgoing ones are displayed on an outer circle.

This method refers to the fact that *incoming attributes* are part of the definition of the concept. *outgoing attribute*s can be chosen to create or visit another concept. The major advantage of the inner-outer representation is that for placing attributes, the whole circle can be used.

Top-Bottom Representation An alternative representation is to show *incoming attributes* on the upper semicircle and outgoing ones on the lower one. A gap separates the upper semicircle from the lower one.

This method emphasizes on the reading direction of lattices which is from top to bottom. Relations to other concepts can be easy interpreted. Another important benefit of this method is that is works better when considering a hierarchical structure of facets and attributes as shown in Figure 6.22. A inner-outer representation has no space for showing a hierarchy.

A major drawback of this method is that *outgoing attribute*s can only be laid out at

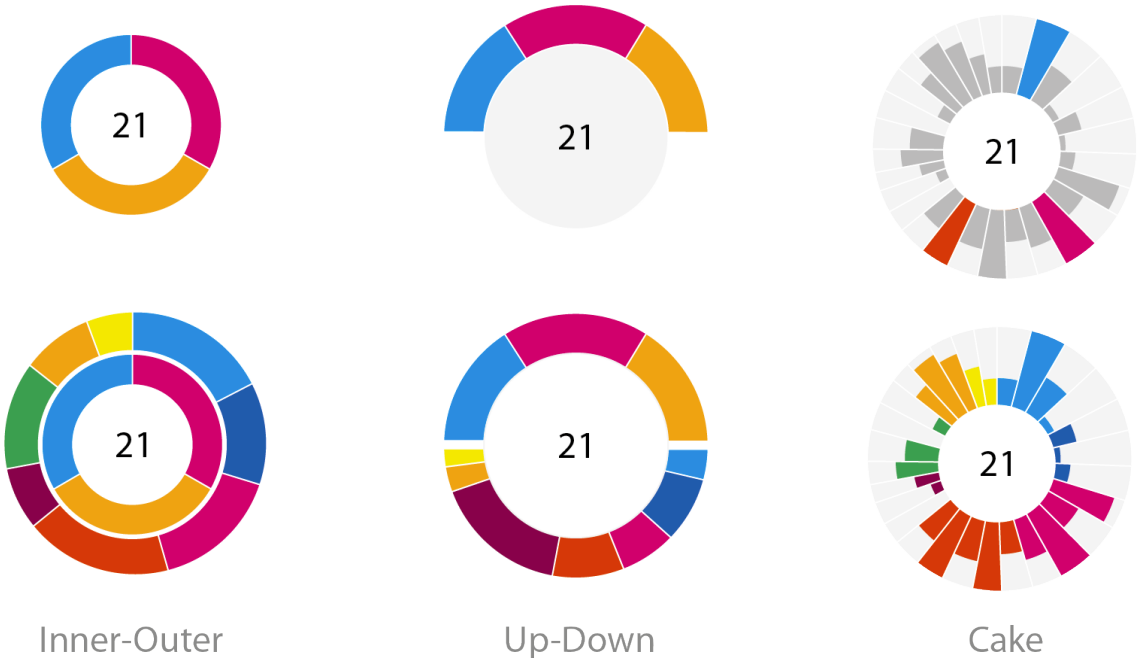


Figure 6.20: Three types of representing a concept with *incoming* and *outgoing attributes*. Circle segments represent attributes and colors indicate their facets.

half the space as with the inner-outer representation.

Cake Representation This representation does not distinguish between *incoming* and *outgoing attributes*. All attributes are shown in a cake diagram. The extension from the inner circle to the outer one, represents the amount of objects within the concept that are labeled by the attribute. Circle segments spanning the complete space describe the whole extent and hence are part of the intent.

There are, however, difficulties when an *outgoing attribute* describes almost all objects from the intent. An example are the orange attributes in Figure 6.20.

A general problem among all representations is the over representation of color. Top-bottom shows less color and is not as "nervous" as the cake representation. Also the fact that top-bottom is read from top to bottom makes it favorable in contrast to the other two representation. To additionally lower the impact of color an alternatives to each representation is designed. The upper circles in Figure 6.20 show only *incoming attributes*.

The final concept representation used by *Facettice* is a top-bottom layout for *incoming* and *outgoing attributes*, combined with a cake diagram for showing distribution. Figure 6.21 shows a concept with three attributes in the intent. The hierarchy among facets and attributes in the example comprises three facets on the top level as indicated by the inner ring. The top level facets together have seven sub facets.

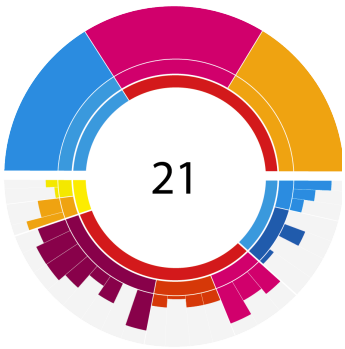


Figure 6.21: Concept node of the Big Fat Lattice. Attributes are colored according to their facet. A value and hierarchy hierarchy is represented by a radial hierarchy.

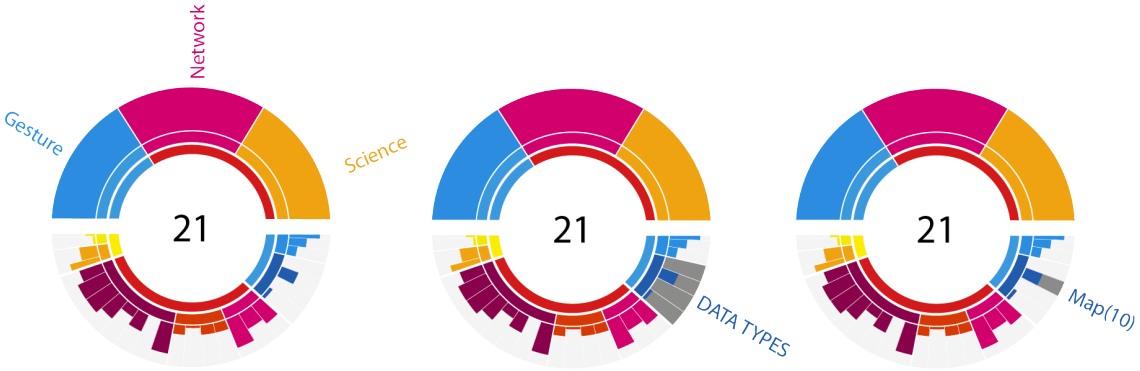


Figure 6.22: The three interaction states of the concept node, *selected* on the left, *facet hovered* in the center and *attribute hovered* on the right.

6.4.3 Facet and Attribute Hierarchy

The realization of a facet and value hierarchy within a top-bottom concept representation is illustrated in Figure 6.22. A top-bottom representation makes it easy to create a radial hierarchy. The radial hierarchy does not restrict the hierarchy depth. Figure 6.22 also shows the final appearance of a concept of the BSL.

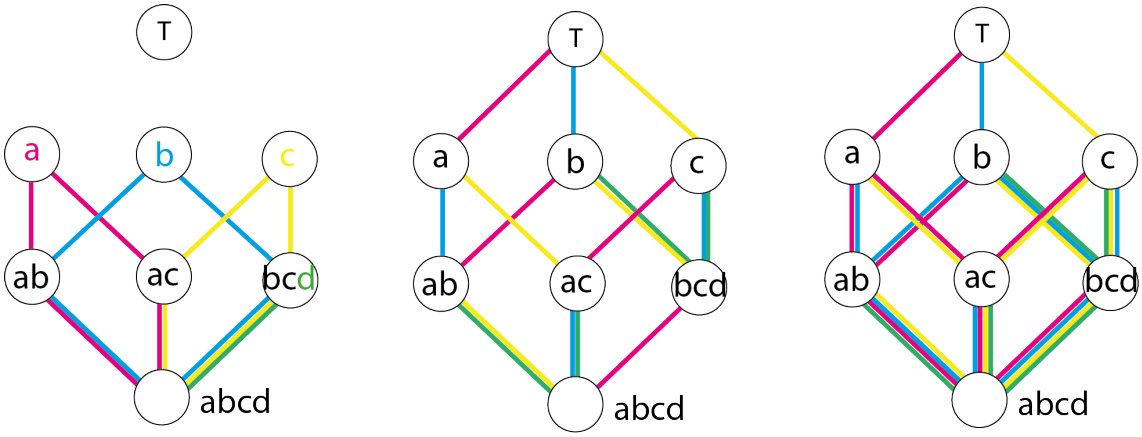
The color coding for the hierarchy is the same as used in the facet and value hierarchy tree component (cf. Section 6.2.5).

Figure 6.22 shows an extended version of the concept node, when the user hovers circle segments. The first concept node shows the name of *incoming attributes*. In the second concept, the user hovers a facet segment and in the last one she hovers a facet value segment.

6.4.4 Relation Representation

As discussed in Section 6.3.1, sub concept relations between concepts can be used to codify different information such as support and confidence. This section presents approaches for coloring relations.

Three different ways to color relations in order to show attribute influence in the



(A) First Appearance (B) Stepwise (C) Complete

Figure 6.23: Three different representations to apply color to attributes. Each attribute is represented by one color.

lattice are developed and evaluated. The influence of an attribute *a* within a concept lattice is shown by all concepts where *a* is part of the intent.

The task is different from coloring relations and concepts in facet lattices (cf 6.3. While a facet lattice shows the influence of an attribute on demand, the BSL must use colored lines by default. Figure 6.23 presents three approaches to color relations, independently from a selected concept.

First Appearance A colored path starts from the most general concept the attribute occurs as shown in Figure 6.23. It leads to the bottom concept by passing all concepts where the attribute is part of the intent. Attribute *d* implies *b* and *c*, hence the path for *d* starts at the concept labeled *bcd*.

Step-wise This method shows only the difference in attributes between two concepts. It corresponds to the perception of *incoming* and *outgoing attributes*. An attribute goes from the super to the sub concept, hence is added to the intent of the super concept and forms a new concept. Vice versa, the same attribute is removed from the sub concept and forms the intent of the super concept. In terms of "traveling" the lattice, this method indicates possible ways, comparable to links between web pages.

Complete This methods superposes *First Appearance* and *Step-wise* coloring. In terms of exploring the lattice, this method corresponds best to the metro map metaphor. A line passes all concepts that are related to an attribute. It also connects all neighbored concepts of these concepts, because they can can be seen as "final destinations".

Although the *Complete* coloring method combines the advantages of the others, it

can not compete with them, because to much lines and colors are shown. *First Appearance* shows well the influence of each attribute within the context by using less lines than the *Complete* method. In comparison to *Stepwise*, *First Appearance* it does not contain confusing "open ends" on lines and creates only enclosed areas.

Step-wise however, is a solution to the problem of many attributes that *First Appearance* also runs into. Since only the attributes are shown that lead to the next concept, no line cluttering will occur. Of course, in the case where many attributes lead to the next concept, multiple lines are necessary, but this is limited to only one concept relation. Regarding the information value of those methods, again *Complete* is weakest due to its delimited readability. *Facettice* uses *Step-wise* coloring by default.

6.4.5 Relation-Concept Join

The last problem in order to create the final BSL visualization is the connection between colored relations and concepts. Last section argued in favor of a *Step wise* relation coloring and Section 6.4.2 developed a top-bottom concept representation. This section discusses ways for combining both representations.

The main argument for a radial top-bottom concept representation is to emphasize attributes as "entry" and "exit" points of a concept. The same reason motivates a step-wise relation coloring. To maintain consitency, colored relations must start at an *outgoing attribute* and end at an *incoming* one. Figure 6.24 shows this idea. The upper concept containing 21 objects has three *incoming attributes* whose relations join the concept at their position on the circle. The *outgoing attribute* `Map` leads to a sub concept that has only 10 objects. The relation starts at the same position on the circle where the segment for `Map` is located. It joins the sub concept at the upper part.

The concept with 15 objects is related to the lower concept by the attribute `Science`. The example shows how a top-bottom concept relation, a step-wise relation coloring and a relation-concept join works together.

The lower concept can be "left" by removing `Map` or `Science`. The other *incoming attributes* in the lower concept can also be removed, and would lead to concepts not shown in the figure. Which concepts are shown in a lattice is determined by the inference algorithm (cf. Section 6.4.1).

A new concept is created when the user clicks at an attribute within a concept. If the user clicks an *outgoing attribute*, a new sub concept is created and the corresponding relation as well. If the user clicks an *incoming attribute*, a new super concept is created, if it does not exist. That way, the user can create new concepts and queries independently from other components of the *Facettice* interface such as the facet and value hierarchy tree.

The position of a relation from a concept to its sub concept can vary over the hole lower semi circle, because the starting position of the relation depends on the

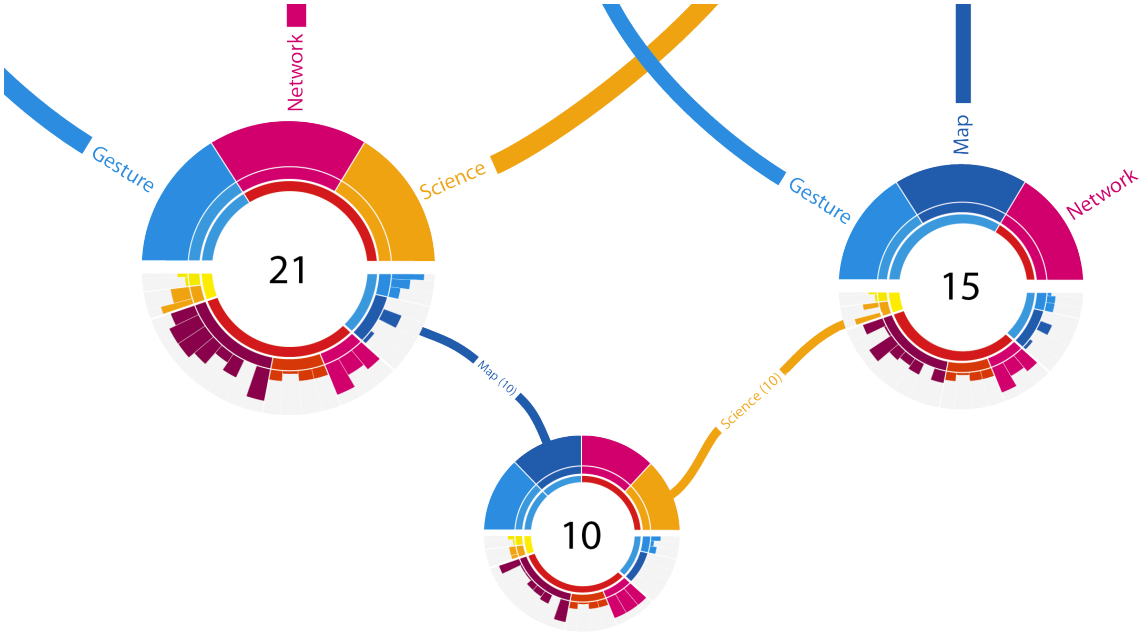


Figure 6.24: Example of a relation-concept join using *stepwise* relation representation. The dark blue relation labeled `Map` indicates that by adding this attribute to the concept on the left, a new concept results with the intent { `Gesture`, `Map`, `Network` and `Science`} comprising just 10 objects. The three relations at the the parent concept lead to three other parent concepts.

position of the *outgoing attribute*. Section 6.4.3 defined that the order and positions of attributes in the concepts is fix. This method assigns a unique angle $\alpha \in [0°, 180°)$ to every attribute within the context. Since every attribute has a unique angle, one can create a vector. A concept lattice can be visualized by placing every concept at the position of the vectors that result from its intent. Lattice A.1 and B.1 in Figure 6.25 show the difference between a justifies and a vector layout.

The advantage of the vector layout is that each attribute is also represented by orientation, which is one of Bertin's visual variables (cf. [Bertin, 1973, p.60ff]). But, the major problem of the vector layout are relations of two attributes as the green-yellow relations demonstrates. To solve this problem, the lattice can be relaxed as shown in Lattice B.2 in Figure 6.25. Relaxing means to calculate a position that satisfies several attributes. In Lattice B.2, two concepts are shifted so that both relations, the yellow and the green one are bended equally. As a side effect, also the blue line is bended.

Still, relations with two attributes do not appear to connect the same concepts. A solution can be *edge bundling* as demonstrated in Lattice B.3 in Figure 6.25 (cf. [Holten, 2006]). Illustration C in Figure 6.25 is a close-up of a concept from Lattice B.3. It shows how an edge bundling can be realized for two arbitrary attributes with angels α_1 and α_2. Point p is the point where both edges are parallel. The join to the sub concept works the same way.

For *Facettice* the vector layout is not usable, because the DelViz Visualization Taxonomy has to many attributes. When only a part of the lattice is created by using the

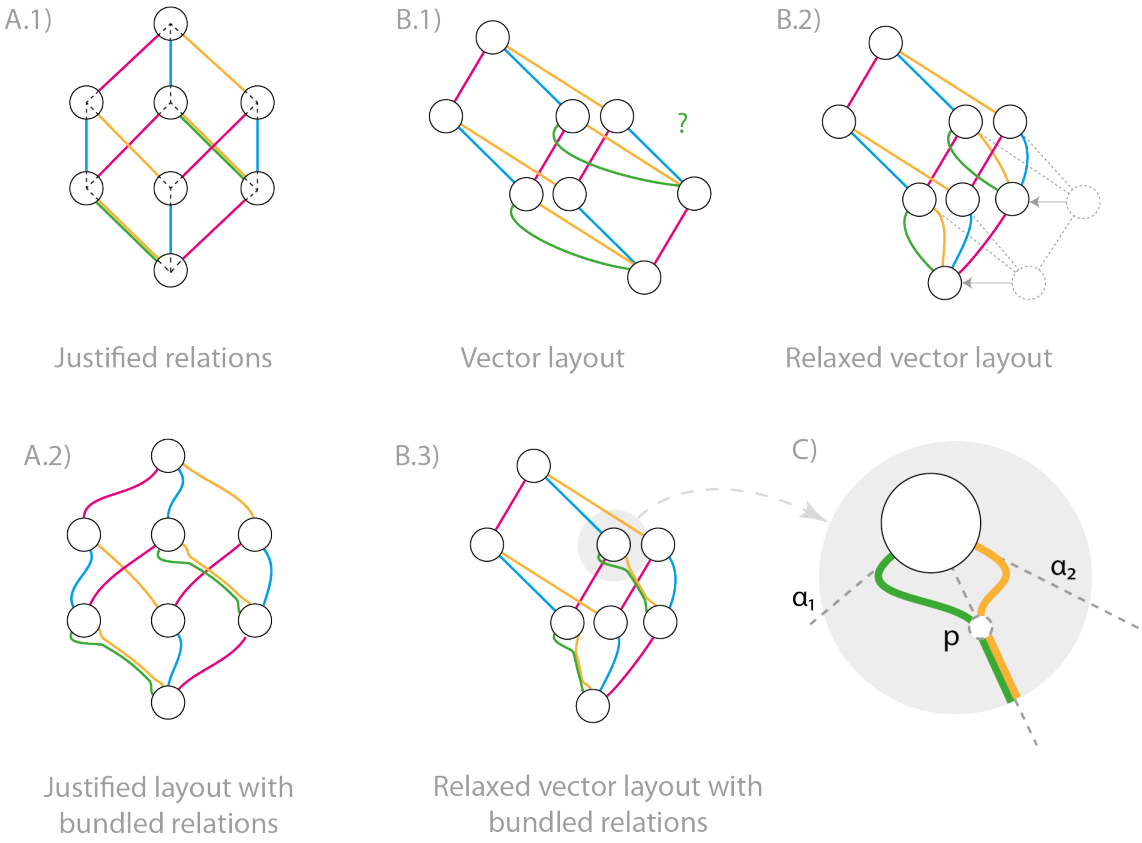


Figure 6.25: Methods and effects on lattice layout and relation appearance for different relation-concept joins.

inference levels explained in Section 6.4.1, concepts are places far from each other. To keep the lattice compact and readable, the justified layout with edge bundling indicated by Lattice A.2 in Figure 6.25 must be used.

6.4.6 History Visualization

The last point to be addressed for an integration of faceted navigation is history visualization (cf. Section 4.7).

A browsing or search history is the ordered set of queries, which a user has posed to the data set. In faceted navigation, he changes his current query successively. Visualizing the history turns into visualizing the order of concepts that corresponds to the users query.

This section first evaluates graphical techniques to show a sequence of concepts within a concept lattice and then considers the structure of a history.

Visualization

The last sections already coded much information into the BSL. Therefore, a history visualization must not interfere with this information. Figure 6.26 shows three simple

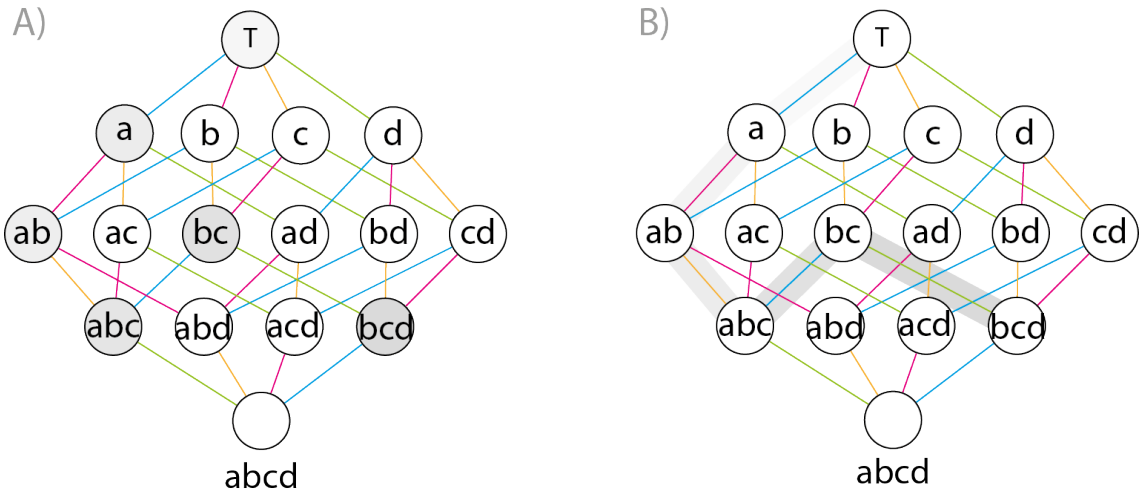


Figure 6.26: Two ways of showing the path the user took while visiting and creating concepts.

methods to visualize a sequence of concepts. The general idea is to show the order by increasing saturation. Saturation is one of Bertin's seven visual variables and can be employed to show an order.

Lattice A colors concepts. Recent visited concepts are saturated, while previous are diminished in their saturation. Emphasizing concepts is confusing if visited concepts are very close. Relations between concepts are already completely employed for illustrating attributes. The background of the lattice serve as an acceptable ground for developing a history visualization since it has not yet been used. It creates a second layer in the background. Lattice B applies the same visual codification to the background of relations. This method reminds more a path and makes it easier to comprehend the history path.

History Structure

Besides a linear succession of related concepts, three *history patterns*, or situations, occur in this user history. Figure 6.27 shows an example history. In Step 1, the user creates or visits concepts until he does a *teleport* to a concept that is not related to the current one. Consequently the path breaks and is continued as in Step 3. By this step the user *re-visits* a previous concept and in the next step he choses an alternative way to a concept that was not visited.

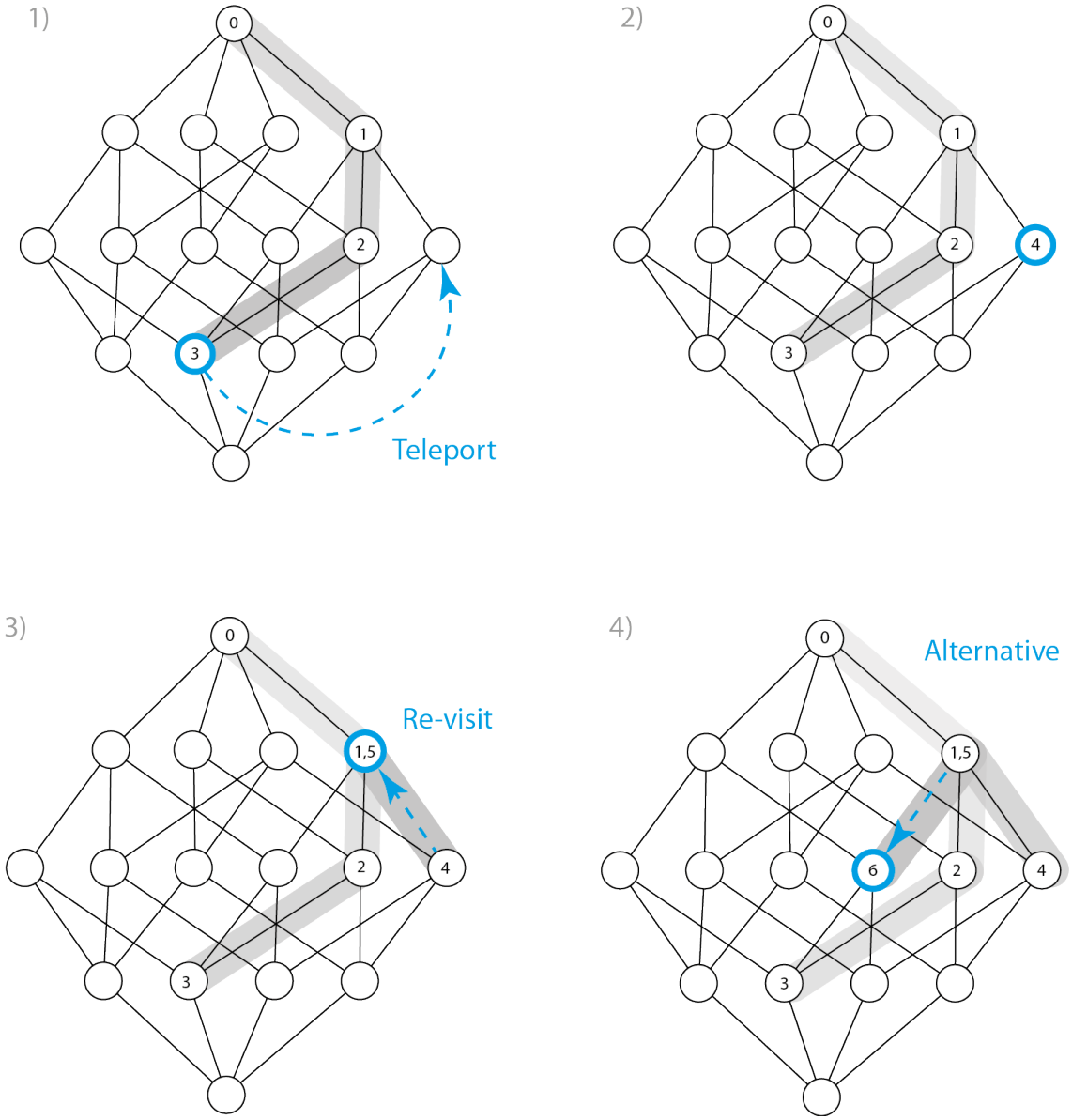


Figure 6.27: Three situations *teleport*, *re-visit* and *alternative*, which occur while visualizing the user hierarchy. The numbers within concepts show the order of visiting.

7 Implementation and Demonstration

Section 7.1 refers to important implementation aspects of the Facettice prototype, such as data retrieval, formal context calculation, lattice layout and general metrics. Section7.2 demonstrates the prototype in respect of the usage scenarios which have been described in Section 5.2.

7.1 Implementation

The prototype of Facettice is provided as an Adobe Air application[1] and runs on almost every platform as ordinary application[2]. Only the Adobe Air runtime environment must be installed[3]. Facettice is written in ActionScript[4] using the Flex[5] framework. This decision is based on the ability of Flex to rapid interface prototyping, event based programming, and good graphical programming possibilities.

The data from the DelViz Visualization Taxonomy is fetched from the Delicious account via HTTP at every start of Facettice. This guarantees full consistency. On client side, code from an open source project *Delicious API*[6] was used as backend to access Delicious.

For calculating the concept lattices, a minor code part from the *OpenFCA* project was reused ([MIT,]). It contains a simple and naive algorithm that calculates the intersections among all formal attributes and puts them in a Spring Graph structure (cf. [Shepard,]). The nodes and their connections represent the concept lattice, which is drawn by Facettice without using any further parts from the Spring Graph (cf. Section 7.1.2).

7.1.1 Implemented Concepts

The purpose of the prototype is to create a preliminary interface to evaluate the primary ideas developed this thesis with real-world data. Facettice is not a compre-

[1] http://www.adobe.com/de/products/air/
[2] Supported are Windows, Mac OS, Linux and Andriod
[3] http://get.adobe.com/air/
[4] http://www.adobe.com/devnet/actionscript/
[5] http://www.adobe.com/de/products/flex/
[6] http://snippets.dzone.com/posts/show/6284

hensive search, browsing and analysis tool. Although, all four interface components described in Section 6.2 are implemented, some minor concepts could not be realized.

The facet and value hierarchy view is used to restrict facets and give feed back about the distribution of facet values. The result representation also shows all objects within the data base and shows additional information on demand. *Facet Lattices* are almost completely implemented and can be used for facet navigation, query preview and adaptation. An combination is missing.

The Big Smart Lattice implements all major concepts like facet and value hierarchy and relation coloring. As a minor part that does not reveal much feedback for a further development of Facettice, the history visualization is not implemented. For Relation-Concept joins, it is used the justified layout with relations starting in the center of the concepts illustrated in Figure 6.25. It is out of the scope of this paper to develop and test different relation-concept joints and edge bundling techniques (cf. Chapter 8).

For successive lattice construction, as presented in Section 6.4.1, no inference is implemented. Higher inference levels require much more calculation and a complete implementation is out of the scope of this thesis. The Big Smart Lattice aims in a first feed back of navigating and constructing a concept lattice. The concept node representation is completely implemented and can be used to navigate from concept to concept and create a concept lattice incrementally. Colored lines show the differing attributes between two concepts and are created using the step-wise method (cf. Section 6.4.4). The lattice layout is adapted each time a new concept is created. The next section explains the used layout algorithms.

7.1.2 Lattice Layout

As reported in Section 2.3, different algorithms and methods can be used. YEVTUSHENKO2004 presents different algorithms and compares them. However, the most important problem in lattice drawing is to minimize edge crossings in order to improve readability and easy pattern detection.

Currently, Facettice uses a *layered approach* to draw the facet lattices and the Big Smart Lattice. It is easier to implement by hand and more flexible in terms of adaptation. A *vector based approach* would be very complicated to implement. Evidently, the treatment of advanced layout techniques is out of the scope of this thesis. Also, the lattice layout is of minor interest for the concepts presented here. The layout is independent from the whole Facettice concept and can be changed later.

To calculate a layout that fulfills the minimized edge crossing criteria, the Barycenter heuristic was implemented (cf [Mäkinen and Siirtola, 2005]). The algorithm is well documented an not explained here in detail.

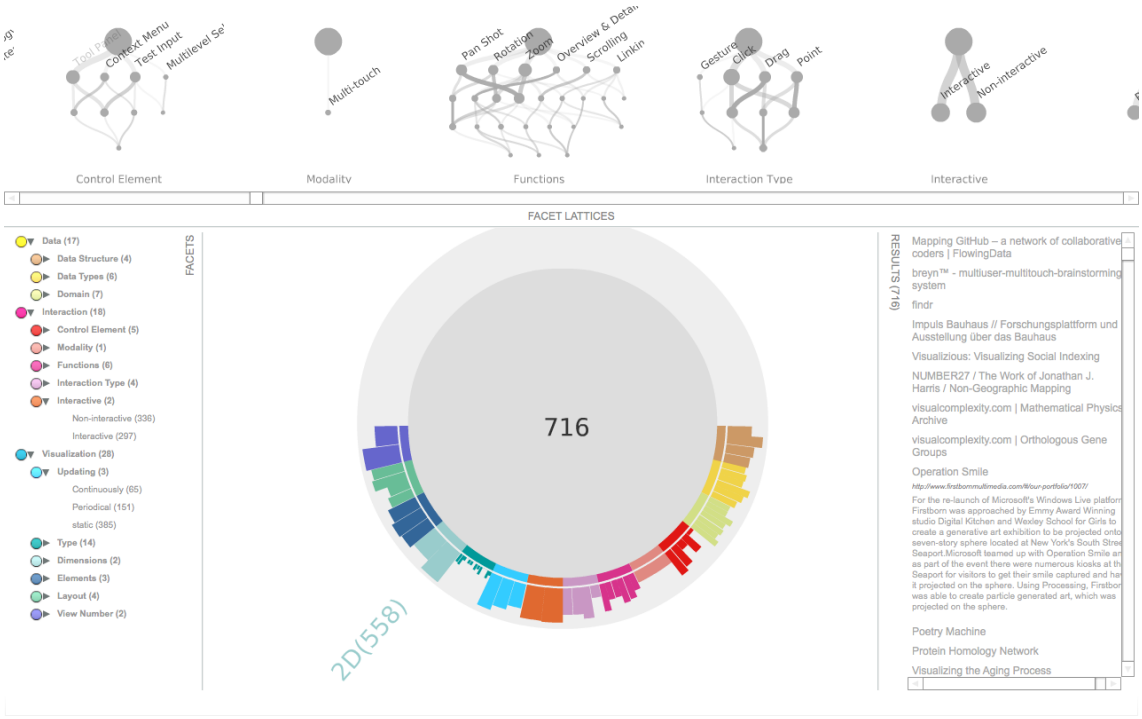


Figure 7.1: Initial screen of Facettice. On the left, the facet and value hierarchy, Facet Lattices on top, the Big Smart Lattice in the center and the lists of results on the right. Within the Big Smart Lattice, the facet value 2D is hovered by the user.

7.2 Demonstration

This section describes the prototype and the created visualizations according to the three scenarios from Section 5.2. Search and exploration are presented together because they use similar functions of Facettice. The second demonstration addresses the analysis of the DelViz Visualization Taxonomy data. However, the focus is to show how Facettice can support data analysis, rather than analyzing the Visualization Taxonomy in detail.

7.2.1 Search and Exploration

This section demonstrates how Facettice supports *Focalized Search* and exploration of the data set. The initial screen is shown in Figure 7.1, presenting the facet and value tree hierarchy on the left, the result representation on the right, Facet Lattices on the top and the Big Smart Lattice in the center.

At the beginning, the BSL comprises only the top concept. It shows at the lower semi circle the facet and value hierarchy and the distribution of facet values. The length of bars shows the amount of objects that are classified under a particular facet value. By hovering the facets and values in the circle, their name is shown. The *Top concept* has no *incoming attributes* because in this particular example, there are no facet values which classify all objects.

Focalized Search

First, the user has to observe the facet hierarchy in order to know which facets and values he can use. In Figure 7.2a the user has selected two values `Interactive` from the facet `Interactive` the and `Continuously` from Facet `Updating` and hence zooms into the data set two times. The Big Smart Lattice created a new concept for each zoom step. The lower one contains both facet values, its super concept only the one that was selected first. On the result panel the user can observe the retrieved objects. Figure 7.1 shows on the right an objects that was clicked to retrieve detailed information.

According to the nature of concept lattices, a zoom-in is directed downwards, while a zoom-out is directed upwards. A shift can be directed in both horizontal directions (cf. Section 4.5).

Assuming the user wants to remove the facet value `Interactive`. He performs a zoom-out from the lowest concept in Figure 7.2a. The two ways for changing the current query are to select or deselect the value within the Dynamic Taxonomy or directly within a BSL concept. Figure 7.2b shows the BSL after the user has clicked the upper semicircle of the outgoing attribute `Interactive`. By clicking an outgoing attribute the user removes this one from the current intent. A new super concept is created above the concept containing all kinds of science projects.

By selecting `Continuously` and then removing `Interactive` the user performs a *Shift operation* (cf. Section 4.5).

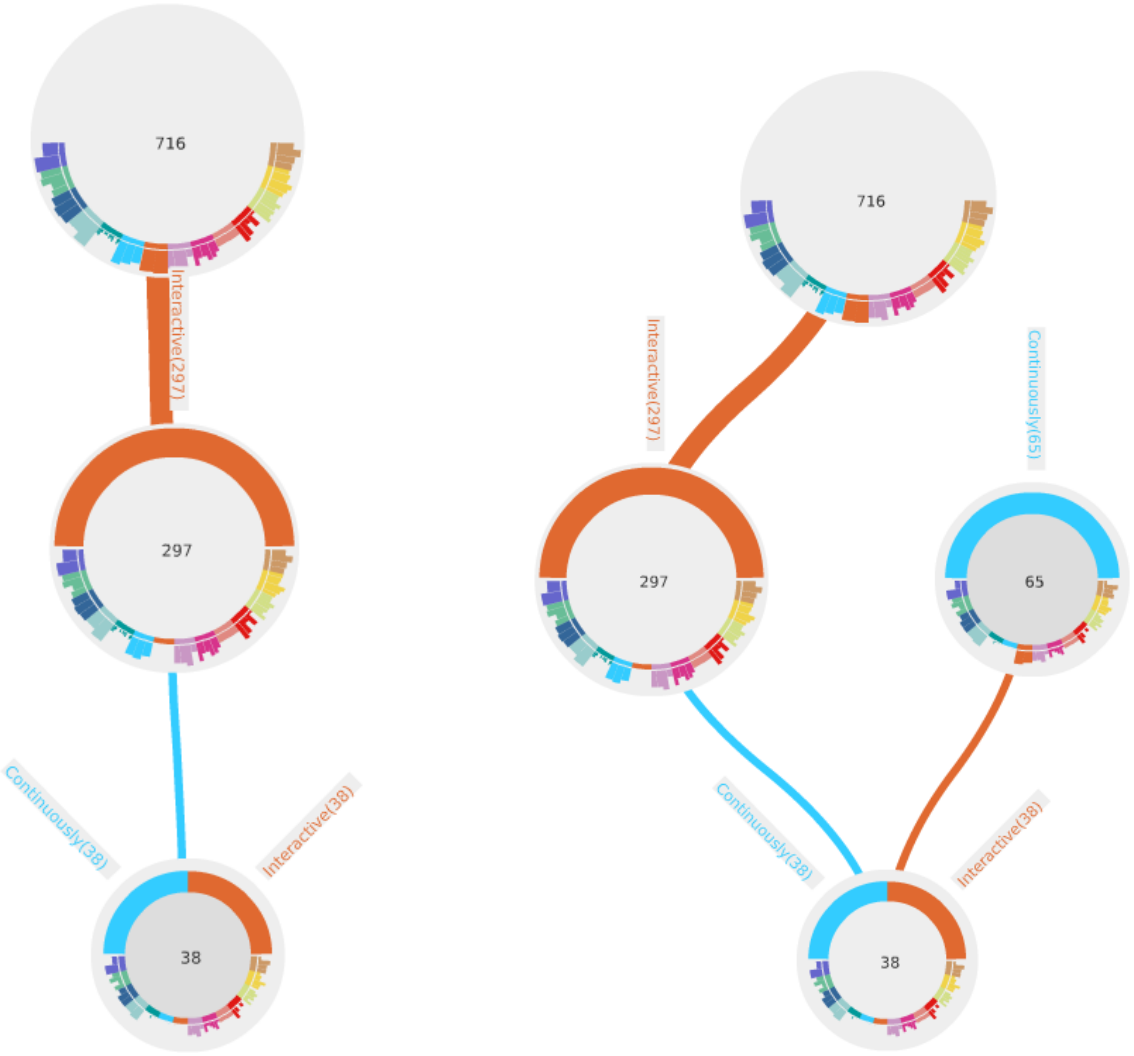
Browsing and Exploration

Now the user starts exploring the data set by further interacting with the Big Smart Lattice. In Figure 7.3a he hovers the outgoing facet value `Multi-touch` from the right concept. By clicking, the user performs a *zoom-in* and a new sub concept is created as shown in Figure 7.3b. The new concept contains the two multi-touch projects that use continuous updating for their graphics.

The step from Figure 7.2b to Figure 7.3b is a *Slice-and-Dice* operation because the user zooms out and then zooms in on another facet value.

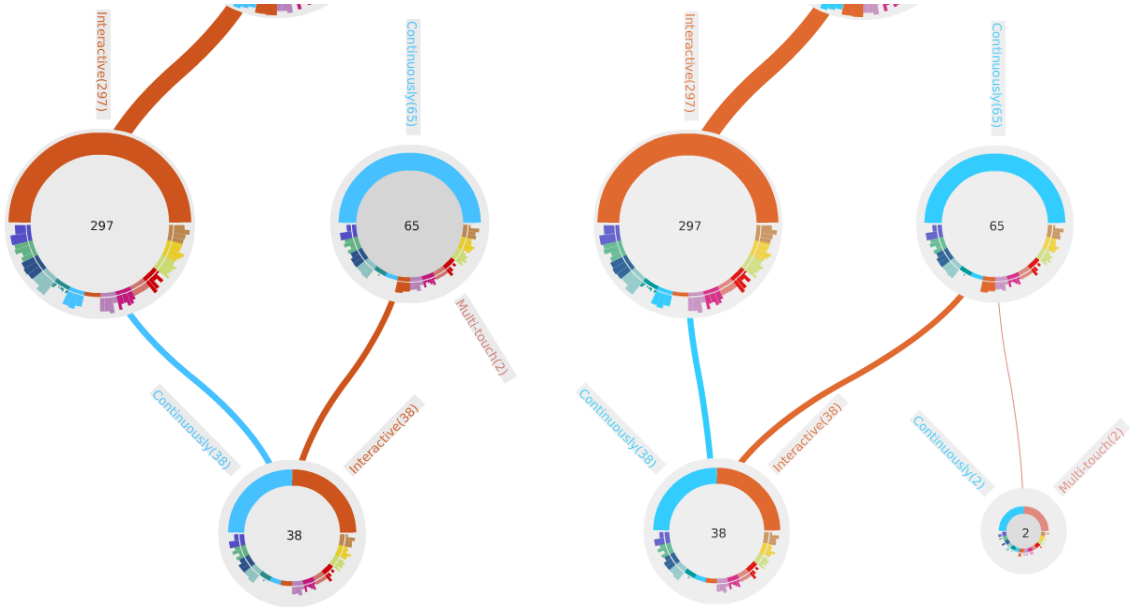
As described in Section 6.3.1, the relations between the concepts differ in their thickness according the *support* of the sub concept. The concept for `Multi-touch` and `Continuously` in Figure 7.3b has a thin line to its super concept, while the concept `Continuously` and `Interactive` is connected by a thicker line to the same super concept. The varying thickness of relations drawn as Bezier curves conveys the perception of *flows*. Bezier curves are a mathematical description of curves with usually one to 2 control points. Those *flows* spread over the whole lattice and indicate clusters and trends.

Figure 7.4 shows a BSL after nine query steps and illustrates the *flows* as well as the layout algorithm used to draw the lattice. At any time, the user can *teleport* to any other concept in the BSL and retrieve its objects. The current focus then is on this



(a) (b)

Figure 7.2: Construction of a Big Smart Lattice. In (a), the user removes the outgoing attribute `Interacitive` and creates a new super concept in Figure (b). The lattice layout is adapted immediately.



(a) Hover `Multi-touch` (b) Create new concept

Figure 7.3: Construction of a Big Smart Lattice. In (a), the user hovers and the attribute *Multi-touch* and selects it in order to zoom-in. In (b), a new sub concept is created, which's intent is {`Continuously`, `Multi-touch`}.

concept and any further navigation is performed from there. This *teleport* causes a non-linear history as described in Section 6.4.6.

By observing the lattice, the user gains a picture of his search and exploration process. He can switch between the queries and see how his queries changed.

Besides changing queries by means of the the Dynamic Taxonomy or within the Big Smart Lattice, the user can also use the Facet Lattices. Their usage is explained in the next section because they are much more related to data analysis.

7.2.2 Data Analysis

The BSL as used by the browsing process in the last section, can also be used for data analysis. The user specifies facet values and create an individual environment. The lattice can grow large and analyzed after all required concepts and relations are created. If *inference techniques* are applied, adding a new value to the BSL causes parts of the whole lattice to be adapted (cf. 6.4.1).

Lattice Visualization

More powerful for data analysis are Facet Lattices. Figure 7.5 shows four Facet Lattices of different character. Non realized concepts are omitted from the visualization. Section 6.3 already described general analysis criteria for Facet Lattices comprising *amount of concepts, depth of lattice, interrelatedness, concept size, bal-*

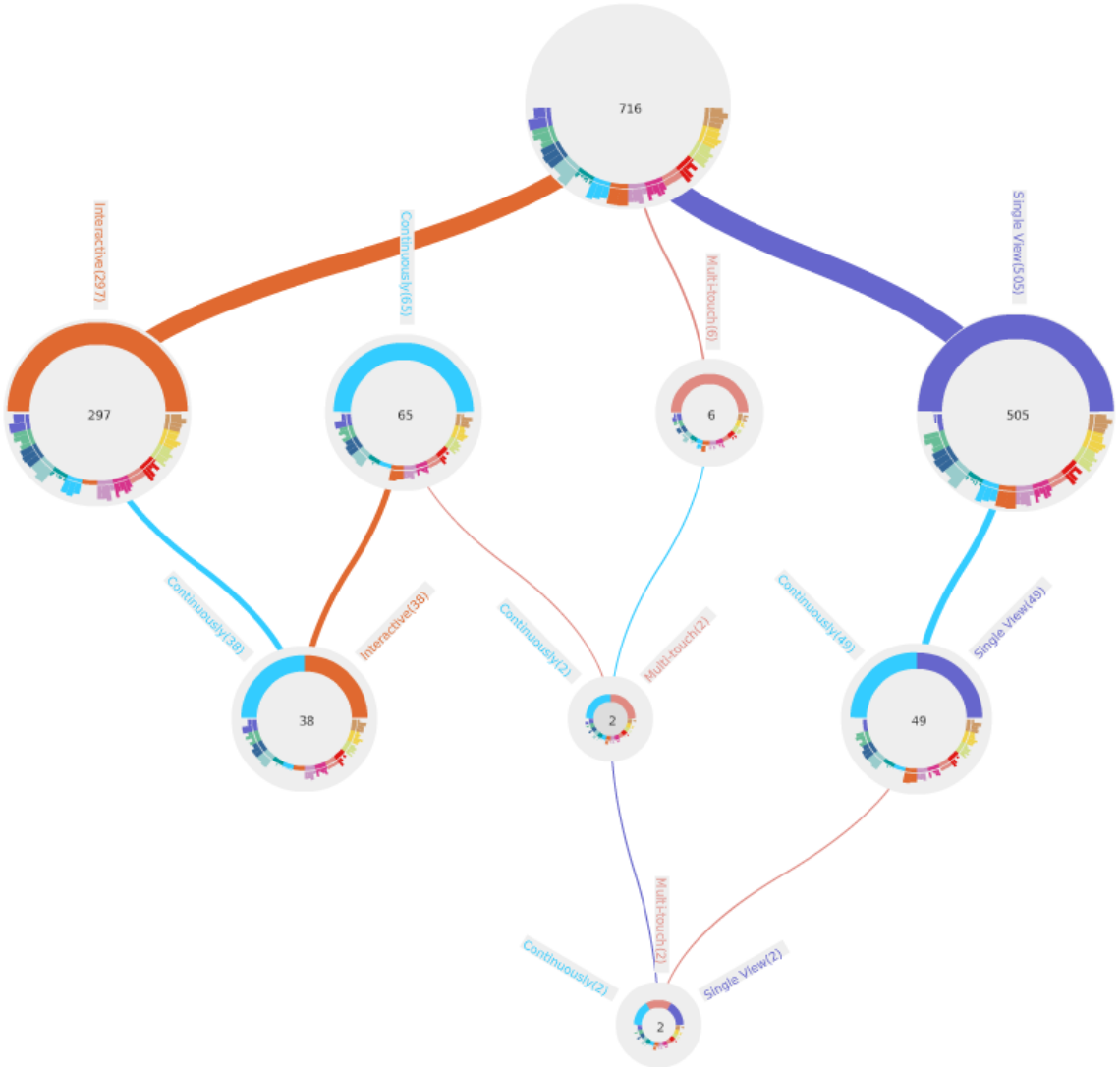
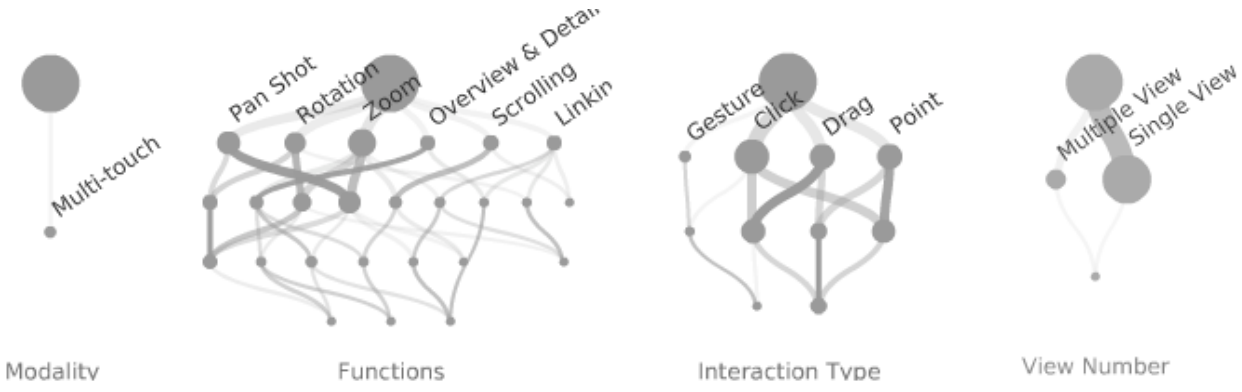


Figure 7.4: Big Fat Lattice after several queries. No inference was used and all concepts are created by the user.



(a) (b) (c) (d)

Figure 7.5: Four Facet Lattices of different realization.

ance of clusters, correlation and *implication*. The facet `Modality` in Figure 7.5 has only one value that is not used very often in the data set, indicated by the small concept and the thin relation. The facet `Functions` has more attributes with a very high interrelation. However, there are no more than four facet values used on the same object, because the lattice has only four levels and no attribute implications do occur. It can also be seen a large correlation between `Pan Shot` and `Zoom`. Their common sub concept has a very high confidence in respect to both super concepts.

The same as for facet `Functions` can be said about the facet `Interaction Type`, which has been analyzed in Section 6.3 on Page 80. Important is the fact there are two clusters, one with almost only `Gesture` and the other one with huge correlation among the values `Click`, `Drag` and `Point`. Facet `View Number` is a complete lattice but with a very strong emphasizes on `Single View`.

Attribute Implication

Besides correlation, implications among attributes have been stated important for data analysis (cf. Section 2.3.3). Figure 7.6 shows a facet that shows implication. The attribute `Dialog` implies `Test Input` as well as `Multilevel Selection`. It also equivalent to `Test Input` $\wedge$ `Multilevel Selection`.

Adaptation

Facet Lattices are adapted each time the user changes a query or selects a particular concept from the BSL. Figure 7.7 shows how an adaptation changes the Facet Lattice. Lattice a shows the initial state with the concept `Click` hovered by the user. All its child and parent concepts are highlighted in the color of the facet `Interaction Type`. If the user clicks a concept in a Facet Lattice, the intent is added to the current query. Lattice b is the adapted lattice after the user has clicked the hovered concept. As seen from this two lattices, the adapted lattice is exactly the sub lattice defined by the clicked concept. Since the attribute `Gesture` was made

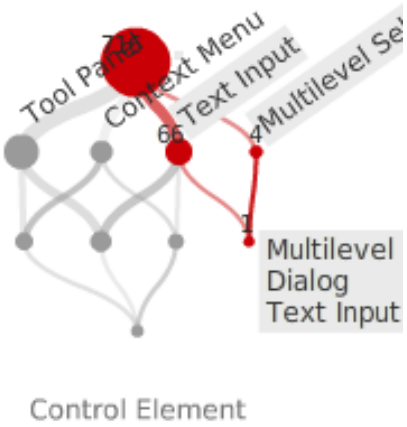
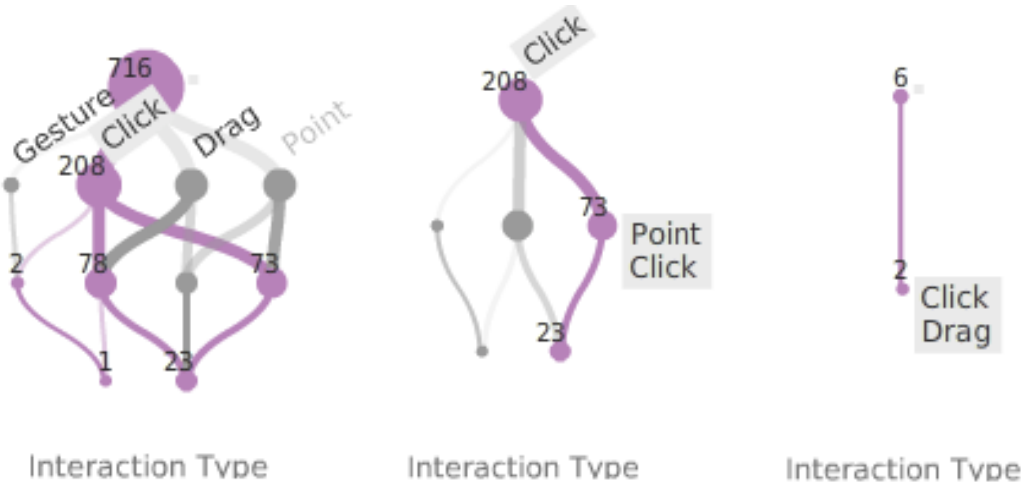


Figure 7.6: Facet Lattice showing an implication among attributes. The lower right concept in red shows the additional attribute `Dialog` within its intent. This attribute does not occur in the intent of the parent concepts. Hence, if an object is labeld `Dialog`, it is also labeled `Text Input` as well as `Multilevel Selection`.

mandatory, all other concepts are pruned.

Figure 7.7c shows another adaptation of the same Facet Lattice. In the current result set, only 6 projects remain classified in this lattice.



(a) Hover `Click` (b) Adapted (c) Adapted

Figure 7.7: Adaption of a Facet Lattice. Figure (a) shows the initial lattice with the concept {`Click`} hovered. In Figure (b), this concept was selected and (c) shows the lattice after another facet was restricted.

8 Conclusion

8.1 Discussion

This thesis presented *Facettice*, a novel way of supporting search and exploration processes by means of interactive concept lattice visualizations and faceted navigation. It therefore relied on four aspects of integrating both technologies *information visualization, navigation in concept lattices, query preview* and *history visualization*.

As could be shown here, interactive *Facet Lattices* provide a quick overview of facets and their realization. They serve for a preliminary data analysis and lowering complexity of the data set by means of facets. So, they can guide the user in choosing the next query and exploring the data set. The *Big Smart Lattice* keeps track of the users interaction and paths through the data. By integrating elements for a selection of facet values directly within the lattice, an integrated interface component could be created. Both visualizations demonstrate the interplay between navigation in data and their visualizations. By that it is aspired to make data more tangible and understandable.

Information visualization is employed to encode information that is important for making query and navigation decisions. It aims in making concept lattices more understandable even to users which are unfamiliar with Formal Concept Analysis. The visualization techniques presented in *Facettice* are: mapping the size of the concept extent, the confidence between concepts, coloring facets and reuse colors within lattices. Also *Facettice* emphasizes related concepts, visualizes a query preview, integrates a facet hierarchy within concepts, distinguishes the intent of a concept from the intent of its child concepts (*incoming* and *outgoing attributes*) and finally designates attributes that make the difference between two related concepts.

Navigation within concept lattices is achieved by two solutions. First, using a concept representation to proceed to an arbitrary concept not shown in the lattice, lowers complexity. Second, navigation is supported by successively constructing the concept lattice. While the first technique makes the user independent from the concept lattice and lets him navigate freely within the data, the second technique creates an exploration environment and preserves the navigation history.

Query preview is realized on two different levels. The lower level consists of numbers and bars associated with each facet value. This technique has already been implemented in common faceted browsers. Because it is a quantitative statement about the extent of every value, this level is one-dimensional and gives only an insufficient picture of the next query. By contrast, concept lattices are two-dimensional because

they show interrelations of facet values for the next and after next query. That way, the user can avoid dead ends or even skip the next step to reach his destination more quickly and precisely. Upon the lattice representation he is also able to make compromises about the results.

The comparison of faceted navigation and Formal Concept Analysis revealed many similarities and particular strengths of each technology. The strengths of both are used as an extension of the respective other. The collection of *interface design patterns* for concept lattices provides a grounded repository for lattice visualization and extensions. This collection needs to be extended and adapted in order to establish a common vocabulary of lattice visualization and its extensions.

The comparison also revealed different kinds of hierarchies, namely intensional hierarchies among facets and facet values as well as extensional ones which are created implicitly from the data. Concept lattices use both: conceptual scales as well as attribute and concept hierarchies. Extensional and intentional hierarchies are an appropriate mean for browsing. While intensional hierarchies present a human-made top down approach to classify data, extensional hierarchies present a bottom-up approach, based on the actual data.

Another aspect that the comparison manifested was that each navigation step in faceted browsing can be interpreted as a change of focus between concepts in a lattice. The concept lattice does not only serve for visualization of data structures, but also as a model for navigation.

The definition of the *Ontological Space of Data* created a common context in which faceted classifications and concept lattice could be related in many ways. First, the model explained the role of FCA as analytical mediator between data and its data model. The realization of data according its data model serves as a particular perspective on the data. The division into regions and aspects as well as the proposition of a visualization process helped to address many particularities of the integration described in this work. The compound of the facet and value hierarchy view, the result representation, *Facet Lattices* and the *Big Smart Lattice* have proven efficient in order to cover as many aspects as possible of the Ontological Space of Data.

The purpose of Facettice and its prototype was not to create a holistic faceted browser or data analysis tool, but rather to develop and to discuss concepts for an integration of faceted navigation and FCA. The development of concepts, and their implementation in particular, aimed to learn more about concept lattices in the context of interactive visualizations. A lot of information for future extensions and modification has been obtained and described.

8.2 Future Work

Some concepts could not have been implemented so far. This include the inference of concept neighborhood, history visualization and a holistic solution for combining and

separating *Facet Lattices*. *Facet Lattices* and probably the *Big Smart Lattice* could be integrated into other existing faceted browsers in the form of a more elaborated widget.

The *Big Smart Lattice* can benefit from alternative representations for concepts such as tree maps or nested line diagrams to show remaining objects and facet values. Chernoff faces, for example, could be used to represent different facet values into a concept representation, instead of using the current representation by the upper semicircle. Attribute matrices could be a possibility to visualize clusters in Alpha Galois Lattices.

Further navigation techniques inside the lattice need to be developed in order to make their construction more intuitive. For now, a concept can only be created but not pruned from the lattice or be combined with other ones. Different input and output media can also influence a further development of *Facettice*. Multi-touch techniques and the implied navigation methods like rotation and dragging are worth a further engagement in this direction. The simple interface of *Facettice*, which uses mostly graphical glyphs for interaction, could be applied to multi-touch displays without larger modifications. Large wall-sized displays urge to deal with big lattices and maintain overview and accessibility.

Multi-touch and wall-sized displays also provoke the question about collaborative browsing and information visualization with concept lattices. Since a *Big Smart Lattice* represents a particular picture to the data, this lattices could be shared, annotated and re-arranged among users and communities.

However, any further development of *Facettice* for browsing and data analysis, requires a detailed user study for the developed concepts. Although the successive construction of concept lattices and the interaction via concepts is not difficult, it must be investigated how users use these techniques. A paper prototype was built for the current conceptualization of *Facettice* at a very low stage of the concept. Its intention was to figure out the general understanding of construction of a *Big Smart Lattice*. It is important to know whether the user relies on the common facet and value hierarchy representation or whether he prefers to interact directly with the lattice as possible in *Facettice*. Another point is up to which size a *Big Smart Lattice* can be used efficiently and whether the user understands the changes caused by an inference and sudden inference of concepts.

In order to drive the process of concept lattice visualization forward, a wider evaluation of FCA software as well as FCA browsers and visualizations seems necessary. Many of these projects are open source and include already many well elaborated algorithms, which could be reused. Also the *interface design pattern catalog* can be extended and patterns described and contrasted.

A last point to address in a future development of *Facettice* is the *Ontological Space of Data*. A first version of the model has been developed by the author of this thesis, in the context of a visualization of Semantic Web data. In this thesis, the model was re-defined and extended to an application of faceted classifications and FCA, but re-

tains its major design. References to ontologies have been made in Section 6.1. The emerging motivation is to proof the current design of the *Ontological Space of Data* and to extend it as well as detail the process that leads to visualizations. Different data sets and structures bear different particularities. Specifically ontologies are a rich source for creating data models as well as expressively describing and relating instances. This motivates a further application of *Facettice* towards a visualization and browsing of large and interconnected data sets from the Semantic Web.

Although data analysis tasks were considered in *Facettice* the major focus remained on supporting search and exploration. However, exploiting the potential of concept lattice visualizations for an advanced data analysis is an open and promising field.

Glossary

Boolean Query	A query whereby different query terms are combined by the boolean operations AND, OR or NOT, 10
EDA	Exploratory Data Analysis, 8
FCA	Formal Concept Analysis, *ger.: Begriffsverband*, 15
Folksonomy	Collaboratively creating and maintaining a taxonomy., 34
Serendipity	Observing or concluding something important without by incindent., 7

Bibliography

[exh,] Exhibit (last checked 8.12.2010). http://www.simile-widgets.org/exhibit/.

[fla,] Flamenco faceted browser (last checked 7.12.2010). http://flamenco.berkeley.edu/.

[pri,] Homepage uta priss (last checked 8.12.2010). http://www.upriss.org.uk/.

[inf,] Infovis wiki (last checked 7.12.2010). http://www.infovis-wiki.net/index.php?title=Linking_and_Brushing.

[lon,] Longwell faceted browser (last checked 7.12.2010). http://simile.mit.edu/wiki/Longwell.

[HCI, 2010] (2010). Hcipatterns (last checked 7.12.2010). http://www.hcipatterns.org/patterns.

[Amar et al., 2005] Amar, R., Eagan, J., and Stasko, J. (2005). Low-level components of analytic activity in information visualization, proceedings of the symposium on information visualization. In *Proceedings of the Symposium on Information Visualization (InfoVis '05)*, pages 111–117.

[Anderson, 2006] Anderson, C. (2006). *The Long Tail: How Endless Choice Is Creating Unlimited Demand*. House Business Books, London.

[Andrienko and Andrienko, 2006] Andrienko, N. and Andrienko, G. (2006). *Exploratory Analysis of Spatial and Temporal Data*. Springer, Berlin, Heidelberg.

[Bates, 1993] Bates, M. (1993). The design of browsing and berrypicking techniques for the on-line search interface. *Online Review*, 13(5):407–424.

[Becker et al., 2002] Becker, P., Hereth, J., and Stumme, G. (2002). Toscanaj: An open source tool for qualitative data analysis. In *Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence ECAI 2002*.

[Bertin, 1973] Bertin, J. (1973). *Sémiologie Graphique - Les diagrammes, les reseaux, les cartes*. Éditions Gauthier-Villars, Paris.

[Broder, 2002] Broder, A. (2002). A taxonomy of web search. In *SIGIR Forum*, volume 36, pages 3–10.

[Carpineto and Romano, 1995] Carpineto, C. and Romano, G. (1995). Ulysses: A lattice-based multiple interaction strategy retrieval interface. In *Blumenthal et al., Human-Computer Interaction*, pages 91–104.

[Carpineto and Romano, 2004] Carpineto, C. and Romano, G. (2004). Exploiting the potential of concept lattices for information retrieval with credo. *Journal of*

Universal Computer Science, 10:985–1013.

[Clarkson and Foley, 2008] Clarkson, E. C. and Foley, J. D. (2008). Augmenting faceted exploration with result maps.

[Dachselt et al., 2008] Dachselt, R., Frisch, M., and Weiland, M. (2008). Facetzoom: A continous multi-scale widget for navigating hierarchical metadata. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI 2008*, pages 1353–1356.

[de Almeida Madeira Clemente, 2010] de Almeida Madeira Clemente, M. (2010). Entwicklung eines visualisierungskonzepts für tag-netzwerke auf einem multi-touch-display. Bachelor Thesis, TU-Dresden.

[Diatta et al., 2009] Diatta, J., Eklund, P., and Liquiere, M. (2009). On concept-lattice applications. *International Journal of General Systems*, 38(4):359 – 362.

[Dörk et al., 2008] Dörk, M., Carpendale, S., Collins, C., and Williamson, C. (2008). Visgets: Coordinated visualizations for web-based information exploration and discovery. In *IEEE Transactions on Visualization and Computer Graphics*, volume 14, pages 1205–1212.

[Ducrou and Eklund, 2005] Ducrou, J. and Eklund, P. (2005). Combining spatial and lattice-based information landscapes. In *Formal Concept Analysis, LNCS*, volume 3403, pages 64–78, Berlin, Heidelberg. Springer.

[Ducrou and Eklund, 2007] Ducrou, J. and Eklund, P. (2007). Searchsleuth: The conceptual neighbourhood of an web query. In *5th International Conference on Concept Lattices and Their Applications (CLA 2007)*, pages 249–260.

[Ducrou et al., 2006] Ducrou, J., Vormbock, B., and Eklund, P. (2006). Fca-based browsing and searching of a collection of images. *Conceptual Structures: Inspiration and Application, LNCS*, 4068:203–214.

[Ducrou et al., 2005] Ducrou, J., Wormuth, B., and Eklund, P. (2005). D-sift: A dynamic simple intuitive fca tool. In Dau, F., Mugnier, M.-L., and Stumme, G., editors, *Conceptual Structures: Common Semantics for Sharing Knowledge*, volume 3596 of *Lecture Notes in Computer Science*, pages 295–306. Springer Berlin / Heidelberg.

[Eklund et al., 2004] Eklund, P., Ducrou, J., and Brawn, P. (2004). Concept lattices for information visualisation: Can novices read line diagrams? In Eklund, P., editor, *Concept Lattces: Second International Conference on Formal Concept Analysis, LNCS 2961*. Springer.

[Eklund et al., 2009] Eklund, P., Goodall, P., Wray, T., Bunt, B., Lawson, A., Christidis, L., Daniel, V., and Olffen, M. V. (2009). Designing the digital ecosystem of the virtual museum of the pacific. In *Proceedings of IEEE International Conference on Digital Ecosystems and Technologies, 2009*, pages 377–383.

[Eklund and Villerd, 2010] Eklund, P. and Villerd, J. (2010). A survey of hybrid

representations of concept lattices in conceptual knowledge processing. In *Formal Concept Analysis, LNCS*, volume 5986, pages 296–311, Berlin, Heidelberg. Springer.

[Elmqvist et al., 2008] Elmqvist, N., Dragicevic, P., and Fekete, J.-D. (2008). Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. In *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis 2008)*, volume 14, pages 1141–1148.

[Erikson, 2010] Erikson, T. (2010). Interaction design patterns page (last checked 7.12.2010). http://www.visi.com/ snowfall/InteractionPatterns.html.

[Ferré, 2009] Ferré, S. (2009). Camelis: a logical information system to organize and browse a collection of documents. *International Journal for General Systems*, 38(4).

[Fincher, 2006] Fincher, S. (2006). Plml - pattern language markup language (last checked 7.12.2010). http://www.cs.kent.ac.uk/people/staff/saf/patterns/plml.html.

[Fincher, 2009] Fincher, S. (2009). Hci pattern-form gallery (last checked 7.12.2010). http://www.cs.kent.ac.uk/people/staff/saf/patterns/gallery.html.

[Gansner and Koren, 2007] Gansner, E. R. and Koren, Y. (2007). Improved circular layouts. In *Proceedings International Symposium on Graph Drawing (GD),*, pages 386–398.

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis*. Springer, Berlin, Heidelberg.

[Garland, 1994] Garland, K. (1994). *Mr. Beck's Underground Map: a history*. Capital Transport Publishing, London.

[Gruber, 2009] Gruber, T. (2009). Ontology. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*. Springer.

[Hearst, 2009] Hearst, M. A. (2009). *Search User Interfaces*. Cambridge University Press, Cambridge, New York.

[Holten, 2006] Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. In *IEEE Transactions on Visualization and Computer Graphics, InfoVis 2006*, volume 12, pages 741–748.

[Jones et al., 2006] Jones, W., Pirolli, P., Card, S. K., Fidel, R., Gershon, N. D., Morville, P., Nardi, B. A., and Russell, D. M. (2006). It's about the information stupid!: why we need a separate field of human-information interaction. *CHI Extended Abstracts*, pages 65–68.

[Kang et al., 2009] Kang, Y.-K., Hwang, S.-H., and Yang, K.-M. (2009). Fca-based conceptual knowledge discovery in folksonomy. In *World Academy of Science, Engineering and Technology*, number 53.

[Keck et al., 2010] Keck, M., Kammer, D., Wojdziak, J., Taranko, S., and Groh, R. (2010). Delviz: Untersuchen von visualisierungsformen durch eine

klassifizierung beruhend auf social tagging. In *Gemeinschaft Neue Medien, GeNeMe'10*, Dresden.

[Keim, 2002] Keim, D. (2002). Information visualization and visual data mining. In *IEEE Transactions on Visualization and Computer Graphics*, volume 8, pages 1–8.

[Keim et al., 2008] Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. (2008). Visual analytics: Definition, process, and challanges. In *Information Visualization - Human-Centered Issues and Perspectives*, pages 154–175. Springer, Berlin.

[Klerkx and Duval, 2007] Klerkx, J. and Duval, E. (2007). Visualizing social bookmarks. In *Workshop Proceedings of EC-TEL '07*.

[Koester, 2006] Koester, B. (2006). Conceptual knowledge retrieval with fooca: Improving web search engine results with contexts and concept hierarchies. In Perner, P., editor, *ICDM 2006, LNAI*, volume 4065, pages 176–190. Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies.

[Kuznetsov and Obedkov, 2002] Kuznetsov, S. O. and Obedkov, S. A. (2002). Comparing performance of algorithms for generating concept lattices,. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216.

[Lee et al., 2006] Lee, B., Plaisant, C., Parr, C. S., Fekete, J.-D., and Henry, N. (2006). Task taxonomy for graph visualization. In *Beyond time and errors: novel evaLuation methods for Information Visualization (BELIV'06)*, pages 1–5.

[Mackinley, 1986] Mackinley, J. (1986). Automating the design of graphical presentations of relational information. In *ACM Transactions on Graphics*, volume 5, pages 110–141.

[Mäkinen and Siirtola, 2005] Mäkinen, E. and Siirtola, H. (2005). The barycenter heuristic and the reorderable matrix. *Informatica*, 29:357–363.

[Marchionini, 2006] Marchionini, G. (2006). Exploratory search: From finding to understanding. *Communications of the ACM*, 49(4):41–46.

[Marchionini and White, 2008] Marchionini, G. and White, R. (2008). Find what you need, understand what you find. *Journal of Human-Computer Interaction*, 23(3):205–237.

[McDonnell and Mueller, 2008] McDonnell, K. T. and Mueller, K. (2008). Illustrative parallel coordinates. In *IEEE-VGTC Symposium on Visualization 2008*, volume 27.

[MIT,] MIT. Conflexplore (last checked 24.11.2010). http://code.google.com/p/openfca/.

[Morville, 2010] Morville, P. (2010). *Search Patterns*. O'Reilly.

[Morville and Rosenfeld, 2006] Morville, P. and Rosenfeld, L. (2006). *Information Architecture for the World Wide Web*. O'Reilly.

[Newman and Landay, 2000] Newman, M. and Landay, J. (2000). Sitemaps, storyboards, and specifications: a sketch of web site design practice. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 263–274.

[Pernelle and Ventos, 2003] Pernelle, N. and Ventos, V. (2003). Zoom : Alpha galois lattices for conceptual clustering. In *Managing Specialization/Generalization Hierarchies, MASPEGHI 2003*.

[Polowinski, 2009] Polowinski, J. (2009). Widgets for faceted browsing. *Human Interface and the Management of Information. Designing Information Environments, LNCS*, 5617:601–610.

[Priss, 2000a] Priss, U. (2000a). Faceted information representation. *Contributions to Proceedings of the 8th International Conference on Conceptual Structures (ICCS 2000)*, Working with conceptual Structures:84 – 94.

[Priss, 2000b] Priss, U. (2000b). Lattice based information retrival. *Knowledge Organization*, 27:132–142.

[Priss, 2008] Priss, U. (2008). Facet-like structures in computer science. *Axiomathes*, 18:243–255.

[Ranganathan, 1962] Ranganathan, S. R. (1962). *Elements of library classification*. Asia Publishing House, Bombay.

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

[Sacco, 2007] Sacco, G. M. (2007). Research results in dynamic taxonomy and faceted search systems. In *18th International Workshop on Database and Expert Systems Applications, DEXA 2007*, pages 201–206.

[Sacco et al., 2009] Sacco, G. M., Ferré, S., and Tzitzikas, Y. (2009). *Dynamic Taxonomies and Faceted Search - Theory, Practice, and Experience*, chapter Comparison with Other Techniques, pages 35 – 74. Springer, Berlin, Heidelberg.

[Sacco and Tzitzikas, 2009] Sacco, G. M. and Tzitzikas, Y. (2009). *Dynamic Taxonomies and Faceted Search - Theory, Practice, and Experience*. Springer, Heidelberg, Berlin.

[Saracevic, 1997] Saracevic, T. (1997). The stratified model of information retrieval interaction: Extension and applications. In *Proceedings of the American Society for Information Science*, volume 34, pages 313–327.

[Schraefel and Karger, 2006] Schraefel, M. and Karger, D. (2006). The pathetic fallacy of rdf. *International Workshop on the Semantic Web and User Interaction (SWUI),*.

[Shepard,] Shepard, M. Spring graph (last checked 7.12.2010). http://mark-shepherd.com/blog/springgraph-flex-component.

[Shneiderman, 1996] Shneiderman, B. (1996). The eyes have it: a task by data

type taxonomy for information visualisation. In M. Burnett andW, C., editor, *Proceeeings of the 1996 IEEE Syposium on Visual Languages*, pages 336–343, Piscataway. IEEE Computer Society Press.

[Shneiderman et al., 1997] Shneiderman, B., Byrd, D., and Croft., W. (1997). Clarifying search: A user-interface framework for text searches. *DL Magazine*.

[Smith et al., 2006] Smith, G., Czerwinski, M., Meyers, B., Robbins, D., Robertson, G., and Tan, D. S. (2006). Facetmap: A scalable search and browse visualization. In *IEEE Transactions on Visualization and Computer Graphics, InfoVis 2006*, volume 12, pages 797–804.

[Sowa,] Sowa, J. F. Ontology (last checked 7.12.2010). http://www.jfsowa.com/ontology/.

[Stasko and Zhang, 2000] Stasko, J. and Zhang, E. (2000). Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *IEEE Symposium on Information Vizualization, InfoVis 2000*, pages 57–65.

[Stefaner, 2008] Stefaner, M. (2008). Elastic lists for facet browsing and resource analysis in the enterprise. *International Workshop on Database and Expert Systems Application, DEXA '08*, pages 397 – 401.

[Stefaner et al., 2009] Stefaner, M., Ferré, S., Perugini, S., Koren, J., and Zang, Y. (2009). *Dynamic Taxonomies and Faceted Search - Theory, Practice, and Experience*, chapter User Interface Design, pages 75 – 112. Springer, Berlin, Heidelberg.

[Stefaner and Müller, 2007] Stefaner, M. and Müller, B. (2007). Elastic lists for facet browsers. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications, DEXA 2007*, pages 217–221.

[Stumme et al., 2002] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., and Lakhal, L. (2002). Computing iceberg concept lattices with titanic. *Data and Knowledge Engineering*, (42):189–222.

[Tidwell, 1999] Tidwell, J. (1999). Common ground: A pattern language for human-computer interface design. http://www.mit.edu/ jtidwell/common_ground_onefile.html.

[Tidwell, 2005] Tidwell, J. (2005). *Desinging Interfaces*. O'Reilly.

[Tukey, 1977] Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesly, Reading, MA, USA.

[Tvaroek et al., 2008] Tvaroek, M., Barla, M., Frivolt, G., Tomsa, M., and Bieliková, M. (2008). Improving semantic search via integrated personalized faceted and visual graph navigation. In *Theory and Practice of Computer Science, SOFSEM 2008*, number 4910, pages 778–789.

[Valtchev et al., 2003] Valtchev, P., Grosser, D., Roume, C., and Hacene, M. R. (2003). Galicia: An open platform for lattices. In *Using Conceptual Structures:*

Contributions to the 11th Intl. Conference on Conceptual Structures, ICCS'03, pages 241–254.

[van Wijk, 2005] van Wijk, J. J. (2005). The value of visualisation. In *IEEE Visualization, VIS 2005*, pages 79 – 86.

[Welie, 2010] Welie (2010). Welie - patterns in interaction design (last checked 7.12.2010). http://www.welie.com/patterns/index.php.

[Wille, 1999] Wille, R. (1999). *Conceptual Landscapes of Knowledge: A Pragmatic Paradigm for Knowledge Processing*. Springer, Berlin, Heidelberg, New York.

[Yang et al., 2009] Yang, K.-M., Hwang, S.-H., Kang, Y.-K., and Yang, H.-S. (2009). Folksonomy analyzer: a fca-based tool for conceptual knowledge discovery in social tagging systems.

[Yee et al., 2003] Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. (2003). Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI'03*, pages 401–408.

[Yevtushenko, 2004] Yevtushenko, S. (2004). *Computing and Visualizing Concept Lattices*. PhD thesis, Technische Universität Darmstadt.

[Yevtushenko, 2006] Yevtushenko, S. (2006). Conexp (last checked 7.12.2010). http://conexp.sourceforge.net/users/.

[Yi et al., 2007] Yi, J. S., ah Kang, Y., Stasko, J., and Jacko, J. (2007). Toward a deeper understanding of the role of interaction in information visualization. In *IEEE Transactions on Visualization and Computer Graphics, InfoVis 2007*, volume 13, pages 1224–1231.

[Zhang and Marchionini, 2005] Zhang, J. and Marchionini, G. (2005). Evaluation and evolution of a browse and search interface: Relation browser++. In *Proceedings of the 2005 national conference on Digital government research*, pages 179–188.

[Zhou and Feiner, 1998] Zhou, M. X. and Feiner, S. K. (1998). Visual task characterhsization for automatd visual discourse synthesis. In *presented at Conference of HUman Factors in Computing System (CHI '98)*, pages 392–399.