Facet-like Structures in Computer Science [⋆]

Uta Priss

Napier University, School of Computing,
u.priss@napier.ac.uk

Abstract. This paper discusses how facet-like structures occur as a commonplace feature in a variety of computer science disciplines as a means for structuring class hierarchies. The paper then focuses on a mathematical model for facets (and class hierarchies in general), called formal concept analysis, and discusses graphical representations of faceted systems based on this model.

Keywords: conceptual structures, facets, formal concept analysis, hierarchy, lattice

1 Introduction

The aim of this paper is to consider facets from a slightly more interdisciplinary viewpoint with particular attention to how facet-like structures are used in computer science. 10 years ago the notion of "facets" was mainly restricted to library and information science (based on Ranganathan's (1945) work) and to a small subset of sociological research (due to Guttman's (1954) influence). The reason for this is probably because classification research was also mainly confined to such disciplines. Recently, however, an interest in classification has developed in several areas of computer science because of developments on the WWW and because of the increase in the size of data repositories. For example, object-oriented technologies often require large class hierarchies; object-relational databases combine class hierarchies with traditional relational databases; formal ontologies usually contain a "taxonomy" which is essentially a class hierarchy. As class hierarchies reach a certain size, a need arises for reducing complexity and for some type of structuring that can be applied to the class hierarchies. Thus the principle of "facets" is occurring in an increasing variety of information processing disciplines, but not always under the same name. The techniques that have been invented for structuring class hierarchies are often very similar to techniques used for faceting in information science. The next section of this paper discusses a variety of such facet-like structures. The third section introduces formal concept analysis (Ganter & Wille, 1999) as one example of a mathematical model for representing class hierarchies and facets. The last section discusses tools for visually representing and navigating faceted systems based on formal concept analysis.

[⋆] This is a preprint of a paper published in Axiomathes, 18, 1-2 (2008), DOI: 10.1007/s10516-007-9023-y ©Springer Verlag.

2 Facets in different disguises

As mentioned in the introduction, ideas similar to "facets" occur in many disciplines and contexts. In the widest sense, facets are a means for structuring class hierarchies. In library and information science, the notion of "facet" usually refers to subdivisions of a class hierarchy based on re-occurring features, for instance, geographic and temporal features. For example, a book could be classified as "fiction/UK/19th century" using the three facets "topic", "geography", and "time". Within a single facet, as a general rule at most one value can be chosen. This is called "mutual exclusivity" (Spiteri, 1998). Thus a book can be 18th century or 19th century but not both. Facets must be designed carefully to ensure that all required values are provided. For example, if books exist that span several centuries, then the faceted classification would have to include either explicit values, such as "17th-19th century", and so on, or a general value for "multi-century" (following the principle of "exhaustivity"). All combinations of values from different facets can occur, as is shown in Table 1. Thus, in this example, any topic could be combined with any geographical location and with any time value. If all of these combinations were enumerated, the list could be very long. If the faceted classification was used to order books on library shelves, then the order of the facets would need to be fixed (in a "citation order") and the enumerated list would need to be generated. Therefore the real power of faceted classification is only unleashed in the use of computerised interfaces, because in that case, it is not necessary to decide on a fixed order or to enumerate all combinations. In computerised interfaces, displays can be created ad-hoc, based on user queries (Slavic, in this issue). Thus the classification can be generated in a flexible, non-redundant manner.

Table 1. Three facets

Topic	Geography	Time
fiction	USA	18th century
fiction	USA	19th century
...		
non-fiction	USA	18th century
non-fiction	USA	19th century
...		
fiction	UK	18th century
fiction	UK	19th century
...		
non-fiction	UK	18th century
non-fiction	UK	19th century
...		

When computer scientists are first exposed to the idea of "facets", they sometimes contend that facets are really nothing else but fields in relational databases. There is some truth in this observation, but it also ignores some of the more subtle aspects of the idea of "facets". In the rest of this section, differences and similarities between facet-like structures in computer science and in traditional library and information science are discussed. The following list describes common features of facet-like structures. The list uses a combination of traditional library and information science notions (“exhaustivity”, “mutual exclusivity”, “enumeration”, “consistency” (cf. Spiteri, 1998)) with general computer science and knowledge representation notions (“module”, “independence”, “container”, “viewpoint”) in order to highlight the commonalities across the two disciplines.

- In contrast to traditional classification schemes, which researchers in many disciplines find difficult to reach agreement about, facets are smaller and *not monolithic* and thus easier to negotiate and manage.
- Facets often represent *viewpoints* or *aspects* and can depend on cultural, social or contextual factors.
- The purpose of creating facets is to *structure* or *modularise* classification systems.
- Faceted systems are much smaller in size and more flexible than *enumerated systems*, which list any possible combination.
- Facets often have a *container function*: they summarise and contain relevant information for a particular topic or aspect.
- Different facets are *independent* from each other (i.e., changes to one facet do not affect other facets).
- Different facets are usually *mutually exclusive* (i.e., their content does not overlap); the values within an individual facet are also usually mutually exclusive (i.e., only one value of a single facet can be applied to an instance).
- Different facets *can be combined* in multiple ways (but not all possible combinations of facets need to have practical relevance and listing all possible combinations is usually avoided).
- In contrast to classes, which *aggregate* data, the main purpose of facets is to *compose* data (Priss, 2000b).
- Individual facets are internally *consistent*.
- Individual facets can (but need not) be *exhaustive* (i.e., all possible values are listed).
- Facets tend to employ *hierarchical* structures, although this is optional.

Table 2 shows examples of facet-like structures from a variety of computer science disciplines. The features that are used in Table 2 to differentiate facet-like structures are as follows: *secondary* refers to structures that are derived from other structures, for example, by summarisation or by providing an interface between other structures. Structures can be either (manually) *designed* or *emergent*, i.e. directly derived from data in some application, often in an automatic manner. These two features are meant to be mutually exclusive. Both secondary and emergent structures tend to change as soon as new data is added. The values within a facet can form a *hierarchy* or they can be an unordered list. The facets themselves can also be arranged in a hierarchy. Some structures can be modelled around *temporal and spatial* information. An important distinction is between *extensional* and *intensional* structures. This is based on a philosophical view that any concept denotes a set of objects or instances, which forms its extension, and a set of features, characteristics or attributes, which forms its intension (this is further elaborated in the next section). Extensional structures tend to be closely dependent on data whereas intensional structures are somewhat abstracted from data and more theory-driven. Structures are *exhaustive* if all possible values are explicitly listed.

Table 2. Facet-like structures in several disciplines

facet-like structure	secondary	designed	emergent	hierarchy	time+space	extensional	intensional	exhaustive	discipline
facet		x		x			x	x	information science
database field		x					x		relational database
view (table)	x	x				x	x		relational database
class		x		x		x	x	x	ontology, object-oriented design
aspect		x					x		aspect-oriented programming
scale		x		x			x	x	formal concept analysis
situation			x		x	x			situation theory
context			x		x	x			artificial intelligence
channel	x	x		x		x	x	x	information flow theory

The first row of Table 2 states that facets in traditional library and information science are manually designed and can be hierarchical. Although facets can have the content of “time” and “space”, facets themselves do not usually have temporal-spatial “metadata” (although, occasionally the date of the creation of a facet is recorded, but this is rare). Even though data-driven methods may be used in the construction of traditional faceted classifications (i.e. using the principle of “literary warrant” of a sample of books), the final representation of the facets does not usually include instances. Thus facets are intensional. Facets in library classification are often exhaustive because all possible facet values are explicitly listed - although, for example, if detailed information about geographical content of fiction is desired, an exhaustive listing of all real and fictitious placenames would neither be possible nor desirable.

The first example of facet-like structures from computer science are the fields (attributes or columns) of a relational database table. The schema of a relational database, which determines its fields, is usually manually designed. Each instance (or row) of the table has at most one value in each field. Relational databases and their fields are not usually thought to be hierarchical. If the example in Table 1 was a relational database, a user would not expect the value “fiction” to be part of a hierarchy and to have such subclasses as “drama”, “crime”, “love story”, whereas traditional faceted classifications often do employ such hierarchies. Fields of a relational database are, however, assigned to a “datatype” which may imply some kind of substructure. For example, temporal data is usually expected to be linearly ordered (18th century being before 19th century). Relational database tables contain equal amounts of extensional data (i.e. the rows of the tables) and intensional data (i.e. the columns of the tables), but database fields alone are intensional.

A view in relational databases is a secondary structure that is created by selecting and combing data from other tables. Views are facet-like because they often represent particular viewpoints and have a container function. Views are usually manually designed (by entering a query that creates the view). Because views are essentially tables, they contain both extensional and intensional data.

In object-oriented design, classes are containers for objects (or instances) and for methods which are used to operate on the objects (cf. Atkinson et al., 1989). If a traditional library classification was designed in an object-oriented manner, then the books of the library would be instances of the classes. Furthermore, methods such as "borrow", "sort", "re-shelve" would be assigned to the classes. Information scientists might disapprove of the idea that facet-like structures contain instances because traditionally facets cut across the classes of a classification scheme and are applied to classes in a flexible and independent manner. Thus, if facet-like structures contain instances, how could they be re-used in such a manner? The answer to this question is that the methods and the general schema of an instance are re-used, not the instances themselves. There exists another facet-like structure in object-oriented design which is called "aspect" (Kiczales et al., 1997) and is more similar to traditional facets. Aspects address functionality that cuts across all classes of an object-oriented program. For example, logging (i.e. writing relevant information to a log file at certain points in time) and security routines usually cut across programs and can be more easily encoded if aspects are used instead of traditional class methods. Object-oriented classes are part of a class hierarchy, which can be mono- or poly-hierarchical (i.e. allowing multiple inheritance). Methods are listed exhaustively, instances not, because they are often created on demand during the execution of a computer program.

Scales in formal concept analysis are very similar to traditional facets and are further explained in the next section. The similarity between facets and scales was probably first described by Kent & Neuss (1995). The notions of "situation" (Barwise & Perry (1983) and Devlin (1991)) and "context" (Akman & Surav, 1996) from artificial intelligence are included in Table 2 because they have a structuring and container function and have similarities with viewpoints or aspects. Situations and contexts are not usually manually designed, but instead are derived as descriptions of observed behaviour. They depend on the instances that are observed and usually include "events". Temporal and causal relationships can be established among the situations or contexts.

Last but not least, channels (Barwise & Seligman, 1997) are also facet-like structures. Barwise & Seligman describe a formalisation of classifications which is essentially the same as a concept lattice in formal concept analysis (see next section). In this formalisation, channels are classifications that mediate between other classifications and can be used to analyse the information flow between different classifications with respect to changes in their "local logics". Channels are secondary because they mediate between other classifications. Barwise & Seligman do not describe any automated means for creating channels, thus they need to be manually designed for each set of classifications. Their remaining features (hierarchy, extensional, intensional and exhaustive) become clear in the next section, because they are all standard features of concept lattices.

There are many other notions of facet-like structures, which cannot be included in this small selection, for example in artificial intelligence, “microtheory” and “frames” are facet-like. Facet-like structures in a wider range of disciplines are Ramon Lull’s Medieval wheels (Walker, 1996), modern notions of “genre”, “dimension”, “perspective”, and, perhaps, even “category”. It might be an interesting research topic to cast a wider net and explore the similarities and differences among these structures but that is beyond the scope of this paper.

3 A formalisation of facets with formal concept analysis

Class hierarchies and facets can be mathematically formalised with formal concept analysis (Ganter & Wille, 1999). In formal concept analysis, a binary relation is called a “formal context”. Table 2 is an example of a formal context - although the column “discipline” would need to be transformed into a binary attribute and shall be ignored in the remainder of this section. The rows correspond to “formal objects” and the columns to “formal attributes”. The information that is contained in a formal context can also be represented as a concept lattice. Fig. 1 shows the concept lattice for Table 2 (ignoring the column “discipline”). Each concept has a unique extension and intension. The extension of a concept consists of all formal objects that have all the formal attributes of the concept and vice versa. In the diagram, formal objects are written slightly below the lowest concept to which they belong; formal attributes are written slightly above the highest concept to which they belong. The extension of a concept can be derived by following all lines downwards from a concept and collecting the formal objects on the way. The intension is dually derived by following the lines that lead upwards. For example, the concept in Fig. 1 which has “secondary” attached has the extension “view, channel”. The same concept has the intension “secondary, intensional, extensional, designed”. The top concept has all formal objects in its extension. The bottom concept has all formal attributes in its intension. Thus the top concept could be considered “all” and the bottom concept “nothing”. In many applications the bottom concept is not shown in the diagrams. It should be noted that all relationships displayed in a concept lattice always depend on the formal context from which it is derived. Further explanations and an overview of the use of formal concept analysis in information science can be found in Priss (2006).

Because formal concept analysis is a strictly mathematical formalism, formal concepts can only ever approximate philosophical or intuitive notions of concepts. But overall formal concept analysis serves as a suitable model for an Aristotelian idea of “genus proximum and differentia specifica”, inheritance of attributes and class inclusion. For example, a subconcept inherits all formal attributes from a superconcept. A superconcept contains all the formal objects of its subclasses and possibly some further objects. Tree hierarchies are lattices (if the bottom concept is ignored), such as the right-hand side example in Fig. 2. The notion of “lattice” is more general than “tree hierarchy”, but more specific than “poly-hierarchy” (or what mathematicians call a “partially ordered set”). Poly-hierarchies are more difficult to manage than tree hierarchies. Librarians tend to dislike them because they are not suitable for ordering books on shelves. Some programming languages (for example, Java) also avoid poly-hierarchies

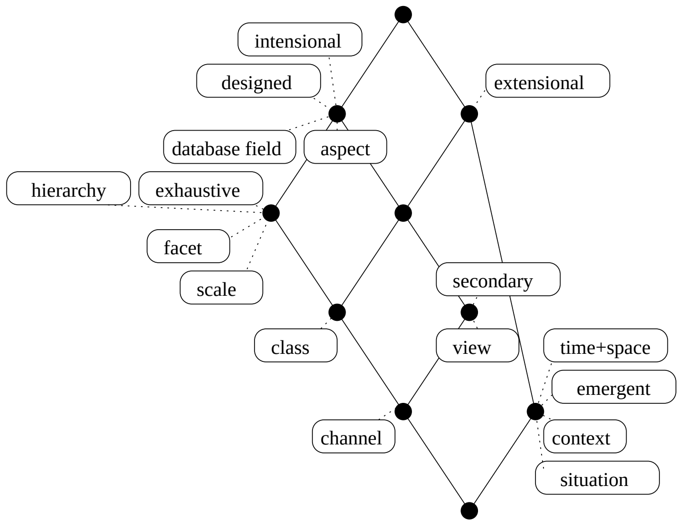

Fig. 1. The concept lattice for the data in Table 2

for management reasons. But because lattices are poly-hierarchies with special mathematical properties, they are easier to search and maintain than general poly-hierarchies. Allowing for each class to have many parent classes means that the depths of a class hierarchy can be kept shallower and in many cases allows for more intuitive representations. Users sometimes find the diagrams of lattices difficult to read, but it is not necessary to either display the whole lattice or even to use graphical representations. In some applications of formal concept analysis, the lattice structure is only used internally, whereas the users browse the information in a more traditional text-based, hierarchical display.

A lattice diagram contains exactly the same information as the formal context from which it was derived. But sometimes "a picture is worth a thousand words". The diagram in Fig. 1 shows that facet and scale have the same attributes in this formal context. Class and channel are more specific than facet: classes contain extensional data and channels contain extensional data and are secondary. Database fields and aspects are more general than facets because they need not be hierarchical and exhaustive. A view is in the same relationship to channel as database fields and aspects are to facets. Contexts and situations have no attributes in common with facets, according to this formal context but they share the formal attribute "extensional" with class, channel and view. An important use of concept lattices is to serve as a tool for scientific debates. The lattice diagram visualises the assertions that were made when the formal context was created. If someone disagrees with any of the relationships in the lattice, then changes need to be made to the formal context. For example, if someone disagrees with the statement that facet and scale are attached to the same concept, then an additional attribute needs to be added to the formal context so that these two formal objects are distinguished.

There is software[1] available that supports this kind of "conceptual exploration" in an interactive manner.

The use of "nested diagrams" (Ganter & Wille, 1999) in formal concept analysis is suitable for representing combinations of facets. Fig. 2 shows two concept lattices which represent two facets. A nested diagram of these two facets is shown in Fig. 3. For simplicity, the bottom concepts of the two lattices are omitted in the nested diagram. For the purposes of nesting, one facet is chosen as the "outer" facet (in this case "literary topic"). Each concept of the outer facet is drawn as a box. A copy of the inner facet is then inserted into each box of the outer facet. Each concept in the nested diagram is a member of two facets. It is possible to nest more than two facets, although it is difficult to visually show nestings of more than three. It would be cumbersome to manually constructed nested diagrams, but a computer program, ToscanaJ[2], is available which automates the process of nesting. A user of ToscanaJ can interactively choose facets (or "scales" as they are called in formal concept analysis) and then navigate through the nested diagrams. By clicking on a node the user will move to a lower level, or, in other words, zoom into a concept. The formal objects can be displayed as counts or as lists. Inner and outer facets in nested diagrams can be switched at the click of a button. ToscanaJ can even be connected to a relational database and can retrieve its formal objects directly from the database.

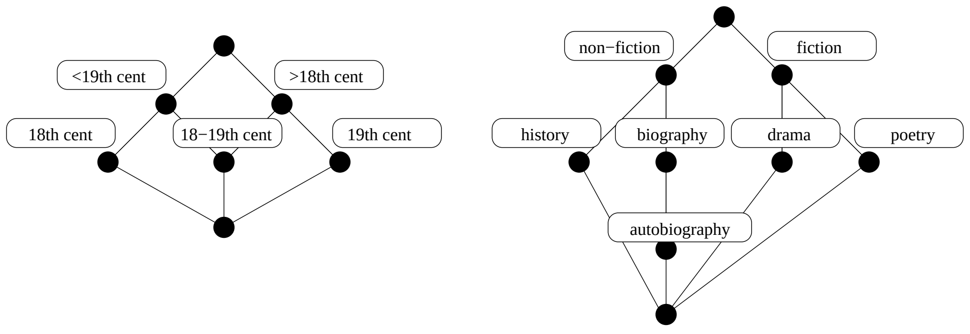

Fig. 2. Two facets: "time" and "literary topic"

In summary, formal concept analysis provides an example of a mathematical model for class hierarchies and facets. Formal concept analysis software can be used to design and navigate class hierarchies and to explore and navigate combinations of facets.

4 Visualisation of faceted systems

The last section of this paper discusses further visualisations of faceted systems. Faceted systems require computerised interfaces for their full power to be unleashed because in

[1] Conexp can be downloaded from: http://sourceforge.net/projects/conexp
[2] ToscanaJ is open-source Java software and is available at http://tockit.sourceforge.net/

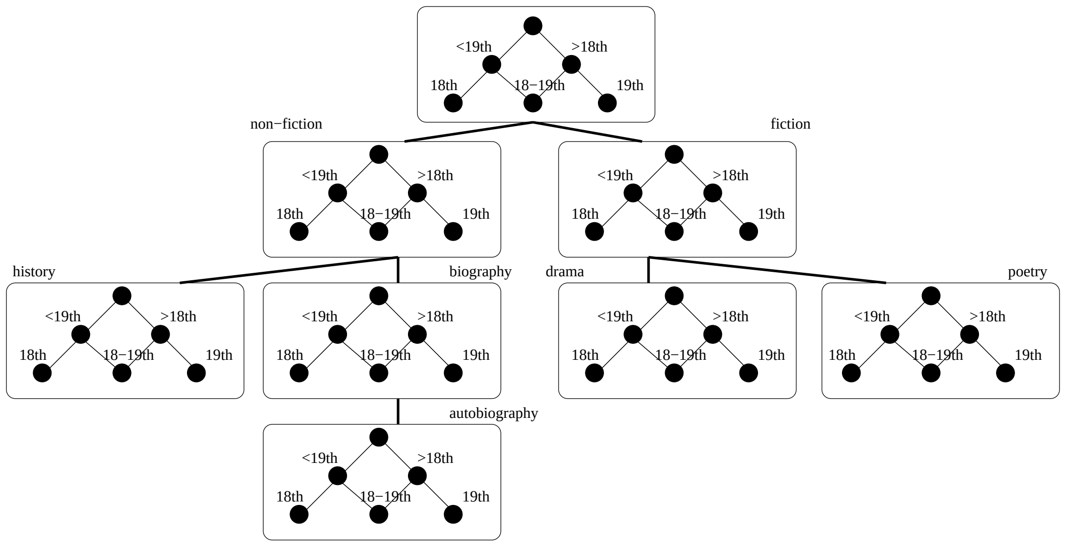

Fig. 3. A nested facet diagram

that way the combinations of the facets can be generated "on the fly" in response to user queries. Interfaces for faceted systems can either be graph-based (as in Fig. 3) or predominantly text-based with a few graphical features (multiple windows, colours, indented lists and so on). It is always possible to show a hierarchy as an indented list so that items on the same level use the same degree of indentation and items at lower levels use more indentation. This is even the case for poly-hierarchies because any poly-hierarchy can be converted into a tree hierarchy by duplicating any nodes that have more than one parent. Text-based interfaces usually provide means for collapsing and expanding the hierarchies. Graph-based interfaces usually provide equivalent means using zooming techniques. For the purposes of this paper, the difference between graph-based and text-based is not important because these are mostly features of the layout, not structural features. If any of these interfaces are implemented using XML then it is, in principle, possible to switch between graph-based and text-based representations. Users often prefer indented list displays because they are very familiar with such displays as they are commonly used by search engines and for browsing file systems.

Interfaces for faceted systems often consist of a collection of windows which show individual facets in a manner that if a user clicks in one window, the information in the other windows changes accordingly as well. Thus the combination between the different facets is often only implicitly shown as a reaction to user actions. For example, in Pollitt's (1998) HIBROWSE system, a user selects a number of facets which are displayed each in its own window. The system then searches for all documents that contain at least one value in each facet. Each value is displayed together with a count of how many documents belong to that value. The total count in all facets is the same because of the condition that all documents relate to at least one value in each facet. A user can click on any value in any of the facets which then reduces the documents in the other facets so that only those are included that contain that value. In addition to the changing document counts, the facets themselves change as well because values with a document count of zero are omitted from the view. In this system the document counts are probably the main guiding feature that helps users to identify their final document set. If the set is too large, a user adds further facets or navigates downwards in the hierarchy of any of the involved facets. If the sets are too small, a user navigates up or deletes facets. The idea of using a faceted hierarchy with document counts as a retrieval interface is also implemented in Sacco's (2006) "dynamic taxonomies" interface[3] and in Hearst et al.'s (2002) Flamenco interface[4].

A system that is similar to HIBROWSE but uses a graph-based display instead of a text-based display is the Faceted Information Retrieval system, FaIR (Priss, 2000a). Fig. 4 shows the two facets from Fig. 3 in a FaIR display. The counts under the nodes indicate how many documents belong to each node. If a user clicks on and highlights nodes in any facet (indicated by the grey areas in the figure), both facets are restricted to documents that belong to one of the highlighted nodes in each facet. In the example of Fig. 3, three of the documents for "19th century" also have the attribute "history"; the other seven have the attribute "autobiography". Priss (2000a) describes the technical details of the FaIR system, the kinds of Boolean queries that should be implemented in such a system and the means for translating the graph-based user actions into such Boolean queries.

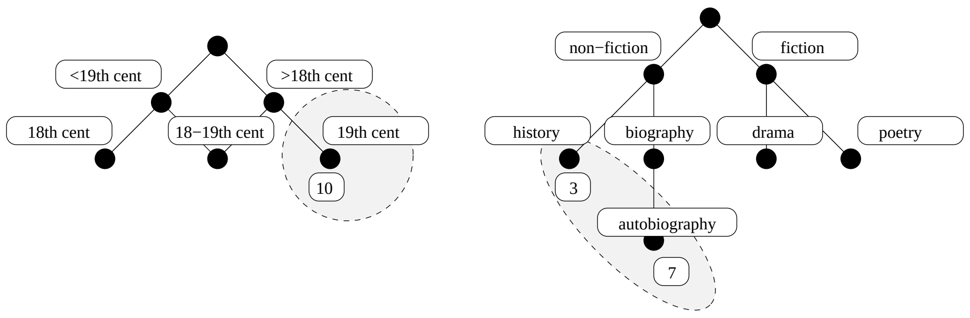

Fig. 4. The FaIR approach: highlighting of Boolean query results

While formal concept analysis traditionally uses nested displays (Fig. 3), FaIR and HIBROWSE use side-by-side displays. The information content is the same in both types of interfaces, but in the side-by-side displays some of the information is only revealed when the user interacts with the systems. It would be impossible to print off a view of all the combinations among the facets in such systems, whereas, at least in theory, such an overview can be printed in a nested display (although the paper might

[3] http://tiziano.di.unito.it
[4] http://flamenco.berkeley.edu

need to be huge and the font tiny). Thus side-by-side displays are more suitable for searching and browsing than for obtaining an overview.

Fig. 5 shows a third, related display method which uses parallel coordinates. Inselberg (1985) introduces parallel coordinates as a means for representing multidimensional vector spaces. Instead of connecting all dimensions of a vector space in a single point of origin and representing instances as points, Inselberg arranges all dimensions in parallel and represents instances as lines that connect the dimensions. Applying Inselberg's idea to formal concept analysis results in a display as in Fig. 5. Facets are displayed in parallel. Each instance (or formal object) is represented as a line connecting the concepts to which it belongs. In Fig. 5, two instances are shown. If all instances were shown simultaneously, the diagram would become fairly messy, but trends might be visible. Thus what was stated about side-by-side displays also applies to parallel coordinates: only part of the information can be shown at any time in response to users clicking and highlighting parts of the facets.

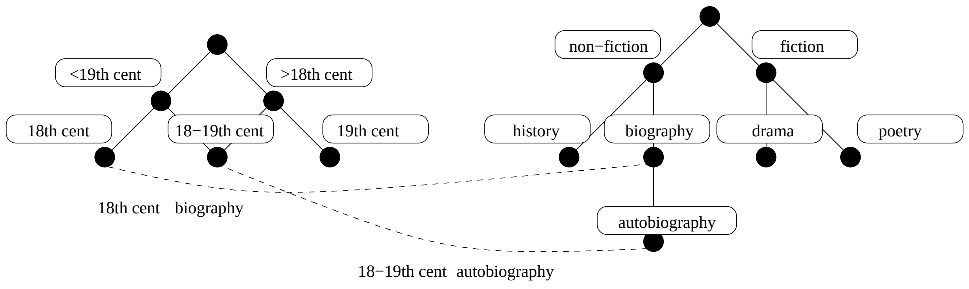

Fig. 5. Facets with parallel coordinates

Some designers of interfaces for faceted systems prefer to use the faceted structures mainly for the internal processing of the system and not as a guiding feature for the interface. For example, Binding & Tudhope's (2004) FACET interface uses the faceted structure much more in the background. Different facets are indicated by using different colours. The interface consists of separate windows for the query, the thesaurus hierarchy and the results. It seems that only one facet hierarchy can be displayed at a time. A user can select any query term and navigate the facet of that term in order to add or delete more values from that facet. When finished with one facet, the user can move to another facet. Additional windows show the definitions of terms or related terms. The results are ranked using a relevance calculation based on similarity measures. Because only one facet is displayed at any time, users will find this interface very similar to interfaces for non-faceted systems which they have encountered elsewhere.

In summary, the modelling of facets with formal concept analysis provides a variety of means for visualising combinations of facets. Facets can be nested or they can be displayed side-by-side, in which case the connections between the facets are indicated by shared instances. To our knowledge, not much research has so far been conducted with respect to the usability of the different types of displays described in this section. It is not yet known whether facets are suitable mainly as internal structures (for structuring class hierarchies) or also more directly for end-user displays, which would require users to have some patience in order to learn how to read such representations.

5 Conclusion

This paper argues that facet-like structures are a commonplace feature of knowledge representation and information processing. Wherever class hierarchies of a significant size are used, facets provide a means for structuring and modularising such hierarchies. Concept lattices in formal concept analysis can be considered a mathematical model of class hierarchies and facets and can be utilised to derive graph-based displays of facet combinations. Ultimately, the search for a "perfect" interface for faceted systems is still ongoing. Interfaces need to keep a balance between the internal use of faceted structures and the external means for presenting these structures to the users because current users are more familiar with predominantly text-based, non-faceted systems. It is hoped that this description of the interdisciplinary nature of facet-like structures and of the different types of displays for faceted systems might stimulate future research into a more widespread use of faceted systems and into means for improving their interfaces for general users.

References

1. Akman, V.; Surav, M. (1996). *Steps toward formalizing context.* AI Magazine, 17, 3, p. 55-72.
2. Atkinson, M.; Bancilhon, F.; DeWitt, D.; Dittrich, K.; Maier, D.; Zdonik, S. (1989). *The Object-Oriented Database System Manifesto.* In: Proceedings of the First International Conference on Deductive and Object-Oriented Databases, Kyoto, Japan, p. 223-240.
3. Barwise, Jon; Perry, John (1983). *Situations and Attitudes.* Cambridge, MA: Bradford Books/MIT Press.
4. Barwise, Jon; Seligman, Jerry (1997). *Information Flow. The Logic of Distributed Systems.* Cambridge University Press.
5. Binding, Ceri; Tudhope, Douglas (2004). *KOS at your Service: Programmatic Access to Knowledge Organisation Systems.* Journal of Digital Information, 4, 4.
6. Devlin, Keith (1991). *Logic and Information.* Cambridge University Press.
7. Ganter, B.; Wille, R. (1999). *Formal Concept Analysis.* Mathematical Foundations. Berlin-Heidelberg-New York: Springer, Berlin-Heidelberg.
8. Guttman, Louis (1954). An Outline of Some New Methodology for Social Research., Public Opinion Quarterly.
9. Hearst, Marti; English, Jennifer; Sinha, Rashmi; Swearingen, Kirsten; Yee, Ping (2002). *Finding the Flow in Web Site Search.* Communications of the ACM, 45, 9, p. 42-49.
10. Inselberg, Alfred (1985). *The Plane with Parallel Coordinates.* Visual Computer, 1, 4, p. 69-91.
11. Kent, R.; Neuss, C. (1995). *Conceptual Analysis of Resource Meta-Information.* Computer Networks and ISDN Systems, 27, p. 973-984.
12. Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Videira Lopes, C.; Loingtier, J.; Irwin, J. (1997). *Aspect-Oriented Programming*. In: Proc. of the European Conf. on Object-Oriented Programming, Finland, Springer, LNCS 1241.

13. Pollitt, A Steven (1998). *The key role of classification and indexing in view-based searching.* International Cataloguing and Bibliographic Control, 27, 2, p. 37-40.
14. Priss, Uta (2000a). *Lattice-based Information Retrieval.* Knowledge Organization, 27, 3, p. 132-142.
15. Priss, Uta (2000b). *Faceted Knowledge Representation.* Electronic Transactions on Artificial Intelligence, 4, C, p. 21-33.
16. Priss, Uta (2006). *Formal Concept Analysis in Information Science.* In: Cronin, Blaise (ed.), Annual Review of Information Science and Technology, 40, p. 521-543.
17. Ranganathan, S. R. (1945). *Elements of Library Classification.* Poona: N. K. Publishing House.
18. Sacco, Giovanni Maria (2006). *Some Research Results in Dynamic Taxonomy and Faceted Search Systems.* SIGIR'2006 Workshop on Faceted Search, Seattle, WA, USA.
19. Slavic, Aida (this issue). *Faceted Classification: Management and Use.*
20. Spiteri, Louise (1998) *A simplified model for facet analysis: Ranganathan 101.* Canadian Journal of Information and Library Science, 23, p. 1-30.
21. Walker, Thomas D. (1996). *Medieval faceted knowledge classification: Ramon Llull's trees of science.* In: Knowledge organization, 23, 4, p. 199-205.