Lattice-based Information Retrieval

Uta Priss

School of Library and Information Science, Indiana University Bloomington,
`upriss@indiana.edu`

Abstract. A lattice-based model for information retrieval has been suggested in the 1960's but has been seen as a theoretical possibility hard to practically apply ever since. This paper attempts to revive the lattice model and demonstrate its applicability in an information retrieval system, FaIR, that incorporates a graphical representation of a faceted thesaurus. It shows how Boolean queries can be lattice-theoretically related to the concepts of the thesaurus and visualized within the thesaurus display. An advantage of FaIR is that it allows for a high level of transparency of the system which can be controlled by the user.

1 Introduction

The prevailing model currently used in information retrieval systems is the vector space model. Although it has proven very useful in many applications, it is limited because of the computational complexity of manipulations in high dimensional vector spaces and the problem that only projections in two-, or possibly three-dimensional spaces can be visually represented. In the 1960's other retrieval models were considered besides the vector space model, such as lattice representations, topological spaces, metric spaces and graph models (Salton, 1968) but they were seen as theoretical possibilities that were difficult to practically implement. This paper revisits one of these models, the lattice model, which has been used in many applications within the framework of a theory called formal concept analysis (Ganter & Wille, 1999) but has not yet been widely applied to information retrieval. The retrieval system, FaIR, described in this paper demonstrates that with modern computational technology, especially graphical representations, and some advancement of the methodology the lattice model is feasible. The main result of this paper is the translation of Boolean queries into lattice representations. This paper does not make any claims as to whether the lattice model is superior to any other models but simply shows that the lattice model is feasible. The main purpose of exploring lattice-based approaches is to increase transparency and user control over an information retrieval system that is not a "black box" to the user.

1.1 Lattices in information retrieval

A first detailed formalization of how to use lattices for information retrieval appears to date back to Mooers (1958). His approach is contained in Salton's (1968) famous book and originally received some attention (Soergel, 1967) but has not been further elaborated in the mainstream information retrieval community. Most of the few, current applications of lattices in information retrieval are based on formal concept analysis

(Ganter & Wille, 1999), which was invented in the early 1980's and relates lattices to object-attribute matrices or document-term matrices in information retrieval. Formal concept analysis applications to information retrieval are similar to Mooers's ideas but have been developed independently.

Lattices are used by Fairthorne (1956), Mooers (1958), Soergel (1967), and Salton (1968) to derive a mathematical formalization of a query (or request) language. If a language consists of a set of primitive terms with Boolean AND as the sole operator, then the resulting set of terms can be represented as a Boolean lattice. For example, ”A AND B AND C” is superordinate to ”A AND B”, ”A AND C”, ”B AND C” in a Boolean lattice. If Boolean OR is added, the possible combinations of terms with AND and OR form what is called a free distributive lattice. The number of elements in such a lattice with n terms, AND and OR grows faster than exponentially: a lattice of 3 terms has 20 elements, a lattice of 6 terms has almost 8 million elements, a lattice of 8 terms has 5.6 $\times 10^{22}$ elements (Sloane, 1999). Adding Boolean NOT complicates this even more.

It can be concluded that, although theoretical results concerning query languages and lattices may be interesting, it is not practical to produce a graphical representation of all possible query terms in a lattice. But it should not be concluded that other lattice representations cannot be useful. As an example, a recently developed system, SWEAR (Davis & McKim, 1999), uses lattices implicitly to improve the ranking of result sets. Text-based representations of ranked result sets of Boolean queries are often ordered based on the number of requested terms that appear in the documents. That implies that all nodes of the Boolean lattice that are at the same level are lumped into one rank. SWEAR changes that by superimposing a linear order on the Boolean lattice that assigns a distinct rank to every node based on user-selected term weights.

1.2 Lattices as conceptual hierarchies or thesauri

Although lattices may not be useful for representing all possibilities of Boolean query terms, they are appealing as a means of representing conceptual hierarchies used in information retrieval systems because of some formal lattice properties. The Galois connection of a lattice applied to information retrieval represents an inverse relationship between document sets and query terms: if more query terms are selected, which means the request is more precise, fewer documents are retrieved, and vice versa. This relationship holds in general for conceptual hierarchies: more general concepts have fewer defining attributes in their intension but more objects in their extension, and vice versa. Therefore lattices have been used successfully for representing conceptual hierarchies in formal concept analysis and for type hierarchies in object-oriented modeling. Besides the Galois connection, lattices are superior to tree hierarchies and poly-hierarchies (or ordered sets), which can both be embedded into lattices, because lattices have the property that for every set of elements there exists a unique lowest upper bound (join) and a unique greatest lower bound (meet). This property is useful in many applications.

Formal concept analysis (Ganter & Wille, 1999) represents conceptual hierarchies as mathematical lattices. Each concept has a set of objects as its unique extension and a set of attributes or characteristics as its unique intension. In information retrieval applications, the documents serve as formal objects and the index terms (descriptors, thesaurus terms) serve as formal attributes (compare, for example, Kollewe et al. (1995)). A document-term matrix can equivalently be transformed into a concept lattice. Figure 1 shows an example. In the lattice diagram, each document is described by exactly those terms that are attached to nodes that are above the document node. Each term belongs to exactly those documents that are attached to nodes below the term node. One problem with this approach is that concept lattices can become fairly large and difficult to generate automatically from the data. Carpineto & Romano (1995) suggest therefore approaches to derive parts of lattices and to use fish-eye view techniques. Godin et al. (1993) represent only the direct neighbors of nodes in a textual interface. The software TOSCANA (Kollewe et al., 1995) facilitates the decomposition of a lattice into smaller lattices that are nested. Users can browse through the lattices by zooming between more abstract and more detailed views.

	Java	C++	CGI	programming language
document 1	X	X		
document 2			X	
document 3		X		X
document 4				X

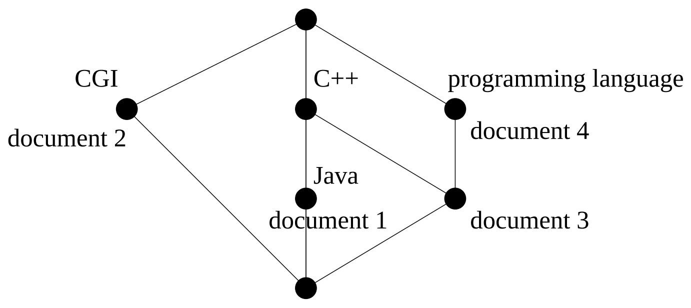

Fig. 1. A document-term matrix and its concept lattice

Most of the applications of lattice theory to information retrieval are data-driven, that is the lattices are constructed from the actual occurrence of documents and terms and not from conceptual relationships among terms that are inherent to the domain knowledge. Therefore, in principle, these approaches face a similar problem to that of the lattice formalisms of the 1960's: all possible combinations can occur and the lattices can become large and complex, although Godin et al. (1993) estimate that the potential maximum complexity is not reached in real applications. Opposed to data-driven approaches are facet-based approaches, which analyze and restrict the possible keyword combinations for each facet; thesaurus-based approaches, which utilize lattices to model the conceptual hierarchy among the concepts; and faceted thesaurus-based approaches, such as the one presented in this paper, which do both.

As an example of a facet-based approach, a pilot study (Rock & Wille, 2000) compiled index terms of a small library with 2000 books into scales, which loosely correspond to facets. The scales are represented as lattices that contain five to ten index terms and all their combinations that can occur among the documents. The scales were manually generated over a period of several months. Using TOSCANA users can browse and navigate through the scales.

Several applications of formal concept analysis to information retrieval utilize thesauri but not faceted thesauri. Priss (1997) discusses several formal methods of combining a document-term matrix with a thesaurus hierarchy. Other approaches (Skorsky (1997) and Groh et al. (1998)) select a subset of a thesaurus hierarchy and generate all possible term combinations of that subset. This produces conceptual structures that can accommodate any document of that domain. But since not all possible combinations actually occur, the approach creates some redundancy. Furthermore, the thesaurus subsets are not usually facets (i.e. conceptually complete and independent). Groh et al. (1998) present a sophisticated method of combining several subsets of a thesaurus hierarchy into one scale. Since the thesaurus is not faceted, two selected subsets of the thesaurus can conceptually overlap. A combination of subsets has therefore to include new terms that correspond to otherwise missing joins of terms from different subsets. The resulting mathematical structure and graphical representation is fairly complicated. If a faceted thesaurus was used instead, the problem would not arise in the first place because facets are by definition complete and independent (compare Priss & Jacob (1998)).

A further lattice-based approach should be mentioned: Pedersen (1993) describes a ”relationship lattice diagram” that consists of a lattice-based thesaurus hierarchy with additional relations. The approach is similar to formal concept analysis but seems to have been developed independently. The resulting diagrams are very interesting but apparently the user interface is still text-based. Furthermore, the embedded lattice is not faceted, the structure is mainly a tree-hierarchy not a poly-hierarchy, and there is no formal explanation of the query process.

All the current lattice-based retrieval models result in browsing interfaces that rely to a certain degree on manually built structures, in contrast to search interfaces based on automatic classification or clustering. Automated retrieval mechanisms as employed in vector space retrieval systems can be applied to lattices if the notions of similarity measure and distance are transferred to lattices. Lengnink (2000) proposes methods of achieving such measures but so far they have not been applied to information retrieval.

2 The information retrieval system FaIR

2.1 An overview of FaIR and its application domain

FaIR is a lattice-based faceted information retrieval system. Before the elements of the system are described, it should be noted that the examples in the following sections are taken from an interface prototype of the Indiana University UITS knowledge base KB (UITS, 1999). The KB is an on-line collection of about 5000 FAQ documents of computing questions. Every document covers one question, such as ”How do I convert between Unix and DOS text files?” with brief explanations and cross-references to related documents. The KB has two interfaces: a hierarchical menu interface and a Boolean search interface. The prototype described in this paper is based on the search interface. The KB was chosen for this study because its document collection is restricted to a well defined domain and fairly homogeneous. The full-text of the documents is automatically indexed by the KB and the query results are ranked. Therefore it is assumed that problems with automatic indexing procedures, word ambiguities and synonyms may hinder some searches. The system, FaIR, described in this paper is currently under development. Once it is established a usability study will be performed to compare the Boolean search interface with the new lattice-based retrieval interface.

FaIR consists of a faceted thesaurus T/F, a set C of concepts that are generated from the thesaurus and a query language Q that is created from concepts and Boolean operators and that is mapped onto sets of concepts using a mapping $L : Q \to \mathcal{PC}$ where $\mathcal{PC}$ denotes the power set of C. Figure 2 provides an overview of FaIR’s components and mappings, which are formally described in the rest of this paper. Documents are represented via a set D of document descriptions that are mapped onto the concepts by a mapping $I : D \to C$. The query language of users is denoted by U and is mapped via $R : U \to Q$ onto the query language Q. The mnemonic for the mappings I, R, L is that I is part of the indexing process, R is part of the retrieval process and L represents the logic of the system. The distinction between query set, document descriptions and thesaurus terms (or concepts) and the mappings in between is based on Salton’s (1968) ideas and has been used in many systems since then. On the other hand, FaIR is distinguished from other systems by its use of a lattice-based faceted thesaurus to generate the concepts and the query language. The graphical representation of FaIR is influenced by TOSCANA (Kollewe et al., 1995), but TOSCANA has not been used for faceted thesauri so far and its display mechanism is different. Therefore, to our knowledge the combination of a lattice-based faceted thesaurus with Boolean queries as described in this paper is a new approach to visualizing information retrieval.

$$\begin{tikzcd}
 & \text{faceted thesaurus} & & \\
 & T/F \arrow[d, no head] & & \\
D \arrow[r, "I"] & C/PC & Q \arrow[l, "L"'] & U \arrow[l, "R"'] \\
\text{document descriptors} & \text{concepts} & \text{query language} & \text{user query language}
\end{tikzcd}$$

Fig. 2. The elements of FaIR

2.2 Mapping document descriptors onto thesaurus concepts

The faceted thesaurus in FaIR consists of a set T of terms that are partitioned into a set F of facets which are lattices. Figure 3 shows an example of two facets. In the left lattice, ”multi-purpose programming language” and ”WWW programming language” have ”programming language” as join and ”Java” as meet. The bottom nodes of the lattices, the meet of all terms in the lattices, are omitted because they are usually meaningless. Every node in a lattice corresponds to a term, which can be a word or a phrase. For single facets, every term (or node) also corresponds to a concept. Compare Priss & Jacob (1999) for further details on the faceted thesaurus formalism used in FaIR.

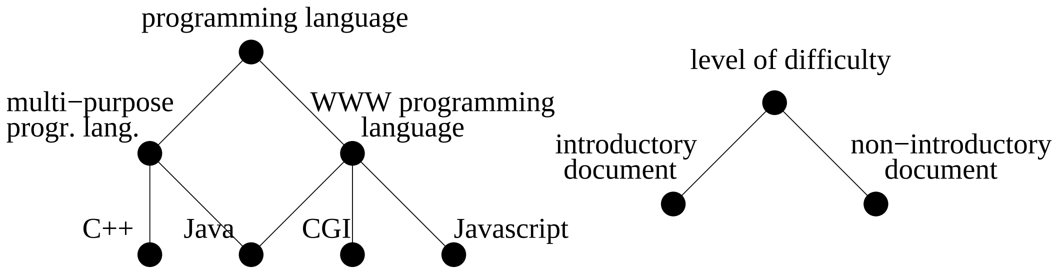

Fig. 3. Two thesaurus facets

For indexing documents, terms from different facets can be combined, such as ”introductory document” and ”Java”. This term composition, which is similar to ”terms with links” (Soergel, 1967), leads to the formation of complex concepts. The set C of concepts consists of simple concepts (single terms from single facets) and complex concepts (term compositions of terms from different facets). Terms within one facet cannot be combined to form concepts because it is assumed that every facet is conceptually complete which means that all necessary combinations are enumerated in the facet. This is not a limitation because facets are restricted to a single viewpoint and are usually small and therefore easy to complete. Ideally the documents are indexed using the concepts of the thesaurus, which means $I : D \to C$ is a one-to-one mapping. It should be noted that this does not mean that documents are indexed by only one term per document but instead that they are indexed by as many terms as needed but at most one term per facet. If the documents are indexed using a different controlled vocabulary, I is a many-to-one mapping and is implemented as a database table that assigns a concept for each document descriptor. If the documents are indexed without a controlled vocabulary or the vocabulary is unknown before retrieving the documents, such as for documents retrieved from the web, I is implemented as a rule set that maps the document descriptors to concepts based on heuristics and/or natural language processing techniques. The rule set that is chosen for I can vary among applications but it is important that it corresponds to L because the performance of the system depends on the appropriate choice of these two mappings.

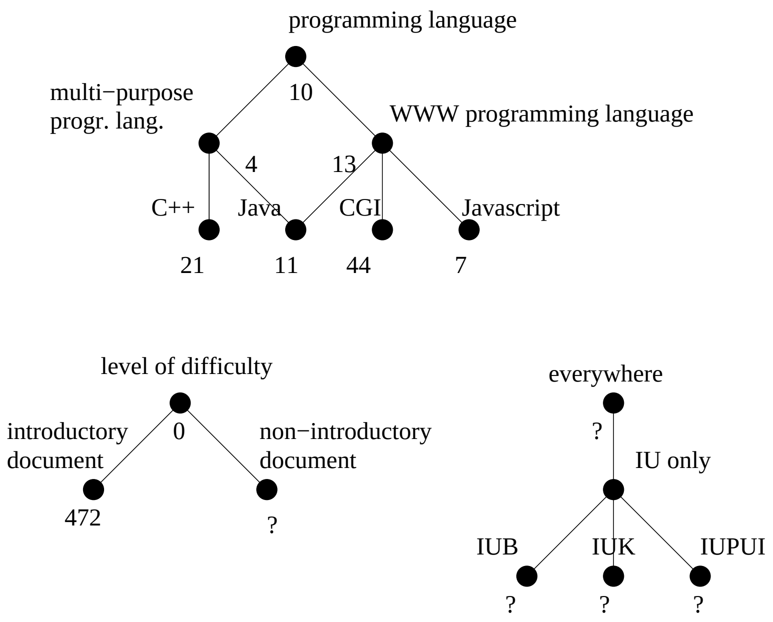

Fig. 4. Thesaurus facets with assigned documents

Figure 4 shows an example of documents of the UITS knowledge base mapped to concepts. The numbers indicate how many documents belong to each concept. A question mark indicates that the number of documents on the concept cannot be determined because of limitations of the current KB interface or that the number of documents is very large (larger than 1000). The facet ”location” (”everywhere” etc) corresponds to a feature of the current KB interface: for every document it is determined whether it is relevant for a general computer community (everywhere) or only for Indiana University (IU only) or for a specific campus (IUB, IUK, IUPUI). This information can only be obtained in combination with a specific topic not with an empty query string hence the question marks. The following rules are used for the mapping I in this application:

- The facets are processed separately.
- A list of synonyms of the terms has been compiled. For the first facet (programming language), only the listed terms are used. For the second facet (level of difficulty), a phrase ”What is” or ”What are” is used as synonym to ”introductory document” because introductory documents in the KB commonly have titles such as ”What is Java”. For the third facet (location), no synonyms are compiled, instead the ”advanced search feature” of the current KB interface is used.
- Documents that contain only one of the terms (or its synonyms) of a facet are mapped to the corresponding concept. This is implemented as a Boolean query for every bottom level concept, such as ”CGI AND NOT (Java OR Javascript OR C++)”.)
- Documents that contain several terms of a facet are mapped to the join of the concepts with the exception that if a document contains both a specific and a general

term, the general term is ignored (see below). This is implemented as Boolean queries for the higher level concepts of each facet, such as ”((CGI AND Javascript) OR (Java AND CGI) OR (Java AND Javascript) OR ”WWW programming language”) AND NOT C++”.)

As an example of the rules, a document on CGI and Javascript (only) is assigned to ”WWW programming language” which is the join of ”CGI” and ”Javascript”. A document on Javascript and Java is also assigned to ”WWW programming language”. The facets of the thesaurus need to be designed carefully so that not too many documents with different descriptors are assigned to the same concept. With respect to the KB, it is not useful to have a concept for the combination of only Java and Javascript under ”WWW programming language”. But for other applications such a concept might be useful. A document that contains ”programming language” and ”Java” but no other terms from the facet is mapped to ”Java”. The more general term ”programming language” is ignored because the document descriptors were derived by full text indexing and many documents start with sentences such as ”Java is a programming language”. Therefore, in this application the more general term often does not add as much information to the document content as the more specific term. In a different application, the rules for mapping descriptors to concepts might be different. For example, manually indexing a document with ”Java” and ”programming language” could indicate that the document is about programming languages in general and uses Java only as an example. This shows that the rules for mapping descriptors to concepts should be formulated only after a careful analysis of the indexing process of the domain.

In this application the documents are assigned to concepts by executing Boolean queries in the current KB interface. A more efficient implementation would pre-process the facets by mapping all documents to the appropriate concepts and then storing document identifiers and concept identifiers in a relational database. The actual numbers would then be produced by issuing an SQL query for each concept. The document counts are only used as an example. Instead of the document numbers, document titles can be displayed. Or the document titles can be retrieved by clicking on the numbers.

Technically every document is mapped onto a single concept not only concerning one facet but concerning all facets. If a document has several descriptors, the descriptors that belong to one facet are mapped onto a single concept in that facet. The concepts of different facets are combined in complex concepts. It follows that although each term belongs to exactly one facet, document descriptors belong to several facets if they represent complex concepts. In that case terms from different facets can have the same synonym. Homographic descriptors must be disambiguated to identify the appropriate facets. This can be done by using natural language processing software or by employing the thesaurus itself for disambiguation by identifying the higher level facets to which a document is mapped. For example, the term ”crane” in a descriptor set {crane, migration, habitat} would point to a different higher level facet than the same term in a set {crane, truck, production}. But word sense disambiguation is a difficult task for any retrieval system and shall not be further discussed in this paper. Concerning the KB, highly ambiguous terms of the domain are stopwords of the system and therefore ignored in documents and user queries. If a single document, such as a conference proceedings volume, covers a variety of topics and mapping it onto a single concept in every facet to which it belongs is not appropriate because the document covers a variety of terms from single facets, the document should be represented as a set of documents which should be indexed separately. But again that is a strategy that applies to any information retrieval system.

2.3 The query language

The query language Q of FaIR is defined as the set C of concepts together with the Boolean operators AND, OR and NOT, i.e. $Q := (C,$ AND, OR, NOT$)$. Elements of Q are called query terms. Each query term is mapped onto a set of concepts via $L : Q \rightarrow \mathcal{PC}$ as described below. The system's internal query language Q is to be distinguished from the query language U of the user because users may not know the exact vocabulary of the system. The mapping $R : U \rightarrow Q$ is based on lookup tables for synonyms and possibly natural language software for word sense disambiguation. It faces therefore problems similar to those of the mapping I because in each case an uncontrolled vocabulary is mapped onto a controlled vocabulary. Since FaIR has a graphical interface, users can browse through the list of facets and search for specific terms of Q. If a user chooses the search interface, the computer checks if the query term exists and is unique. For ambiguous terms, that is terms that are stored in the system with parenthetical information, such as ”crane (animal)” and ”crane (device)”, the computer inquires which one was meant by the user. If the query term does not exist, the computer suggests near matches, such as terms that are alphabetically close. With the browsing interface, users have direct access to Q. In that case, if it is ignored that users may not have the same understanding of the meaning of terms in Q as is intended by the designers of the system, the languages U and Q can be assumed to be equivalent in FaIR.

2.4 Intra-facet searches

The mapping $L : Q \rightarrow \mathcal{PC}$ must correspond to I. Since, in this application, I maps documents with several descriptors to their joins, a search for a single term must also retrieve more general terms. The following applies to L in this application: using Soergel's (1967) terminology, ”exclusive” and ”inclusive” searches are distinguished. An exclusive search retrieves an exact concept. For example, a search for ”Java” retrieves only documents on Java alone but not documents on ”Java and other programming languages”. An inclusive search includes more specific and more general terms because a document on ”programming languages in general” might also be relevant for ”Java”. Formally, an exclusive search for a simple concept retrieves only the documents that are directly attached to that node, or to the concept's nodes in different facets in the case of a complex concept. An inclusive search retrieves all documents that are attached to the concept directly and to nodes below and above the concept. In lattice terminology, an inclusive search retrieves the union of the filter (the nodes above) and the ideal (the nodes below) of a concept. The first example in Figure 5 shows searches for ”multipurpose programming language”. The dashed line indicates the exclusive search while the inclusive search is the area within the solid line curve. In FaIR's interface the results are highlighted using different colors. Users do not have to type queries but can construct them by clicking and highlighting.

The Boolean AND as exclusive search in a single facet retrieves meet and join of the terms. The inclusive Boolean AND in a single facet is represented by retrieving the documents of single inclusive searches for every term and intersecting the resulting sets. The second example in Figure 5 shows a search with Boolean AND. In this case exclusive and inclusive search are identical because there are no further concepts above the join and below the meet of the terms. In general, an inclusive search retrieves the filter of the join, the ideal of the meet and in the case of comparable terms the interval in between. It is a feature of FaIR that general and specific terms are included in the Boolean AND because the mapping I assigns, for example, documents on all programming languages to the top node. Therefore documents on multi-purpose and WWW programming languages can be found at the top node and at the "Java" node depending on whether they are general or specialized documents. In other applications, it may be appropriate to use a mapping L that maps Boolean AND only to the meet and (its ideal) but not to the join. These are design decision for the mapping L.

Boolean OR is represented as a union of documents retrieved by searching for the terms separately (compare Figure 5 for an example). In this application, the inclusive OR is represented as the union of inclusive single searches. The exclusive OR restricts that union to elements between the meet and join. The inclusive OR is probably not very useful because it retrieves too many documents. The exclusive OR on the other hand, shows everything that is related to either one of the requested terms but is not too general or too specific.

Boolean NOT corresponds to the set theoretical difference. Exclusive NOT excludes all documents that are in the ideal of the term to be excluded; inclusive NOT excludes documents in filter and ideal of the term.

2.5 Inter-facet searches

So far the Boolean operators have only been applied to single facets. If several facets are included in one query, it does not seem sensible to use OR between facets. For example, while a query for "Java AND introductory document" is reasonable, a query for "Java OR introductory document" does not correspond to a common sense logical construction because natural language "or" assumes a shared attribute between the terms such as in "green or blue" which share "color". Sensibly applied Boolean OR usually corresponds to synonyms, such as in "car OR automobile OR auto", which belong to a single facet. Therefore in this application, only Boolean AND is allowed between different facets. In inter-facet searches the difference between exclusive and inclusive does not apply to the search as a whole. Boolean NOT is also restricted to single facets because otherwise inter-facet OR's might result according to de Morgan's laws. For example, "Java AND NOT 'introductory document' AND (everywhere OR IUB)" is an acceptable query; "NOT (Java AND 'introductory document')" which is equivalent

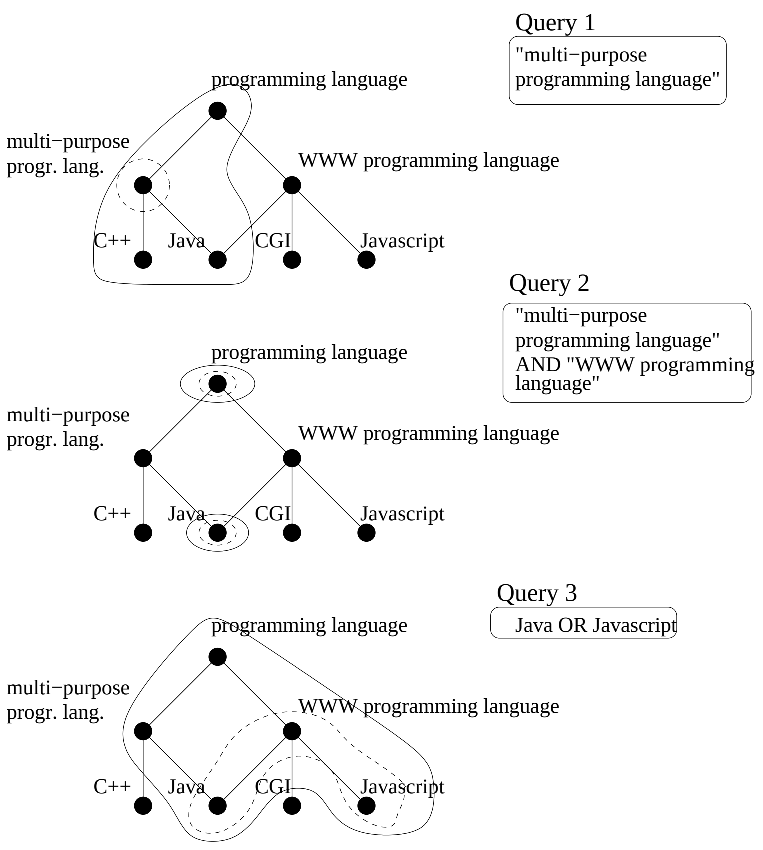

Fig. 5. Several queries in a single facet

to ”NOT Java OR NOT ’introductory document’” is not an acceptable query. As mentioned before, users do not have to worry about these details because they formulate queries by selecting facets and highlighting concepts in these facets.

Figures 6 and 7 demonstrate queries using inter-facet AND. In Figure 6, a user has selected three facets from the KB interface. All terms in all facets are highlighted. This corresponds to inclusive searches for the top nodes of the facets combined by inter-facet AND. The inter-facet AND results in the intersection of the documents of the facets. That means that only the documents that belong to all three facets are counted. In Figure 7, documents on programming languages that are relevant ”everywhere” are selected. Only 65 documents fulfill that condition. The numbers in all three facets are reduced accordingly.

3 Conclusion

An advantage of FaIR is that queries retrieve sets of concepts within the context of conceptual relations. This is in contrast to traditional retrieval systems which show no internal structure of large retrieval sets (except of ranking mechanisms whose functionality is often not clear to the users) or which in the case of an empty retrieval set give no indication as to how the query should be changed to be successful. If too many documents are attached to one node in the retrieval display, users can select additional facets to partition the same set into smaller sets. If no documents are attached to one node, users can identify neighbor nodes that have documents attached. By highlighting certain parts of facets, users can perceive the impact of that selection on related facets and therefore interactively modify the retrieval set until it has an appropriate size. At every point, users have complete control over the system and complete information about the selected facets. Once the result set is small enough, users can click on the document numbers to display document titles, abstracts or the full text of the documents if available.

FaIR’s design is highly modular: the faceted thesaurus is modular in that the facets are conceptually complete and independent of each other. Single facets can be added to or deleted from the thesaurus after an automatic consistency check that assures that terms are not duplicated, links in the facet hierarchy are not missing, and the thesaurus relations are not circular (compare Priss & Jacob (1999)). The thesaurus, set of document descriptors and user query language are connected via mappings. All three can be fairly independent of each other although, if they are totally independent, the system’s efficiency relies heavily on the quality of the mappings. Any faceted thesaurus can be incorporated into FaIR. It follows that users can maintain their own thesaurus as a means of information filtering. In that case users are completely familiar with the query language, i.e. $U = Q$. The only component that might not be totally under user control is the mapping I, although advanced users could change the rules for I manually. The ”black-box phenomenon” of information retrieval systems is thus reduced to natural language processing techniques that can be tested by the user. Users can share their faceted thesaurus or parts of it with other users. They can apply FaIR as a front end to other retrieval systems. It is not suggested that patrons of a library, for example, would be able to use FaIR without some training. The current target user group is

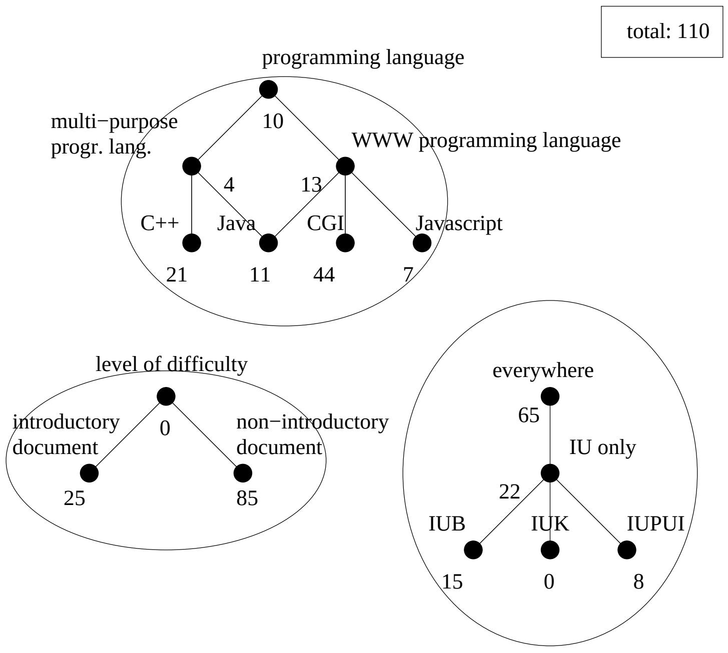

Fig. 6. The query "'programming language' (incl) AND 'everywhere' (incl) AND 'level of difficulty' (incl)"

information professionals that perform queries for patrons and researchers that need to retrieve information concerning a specific domain with high accuracy and convenience and do not mind the effort of learning to use an information retrieval tool.

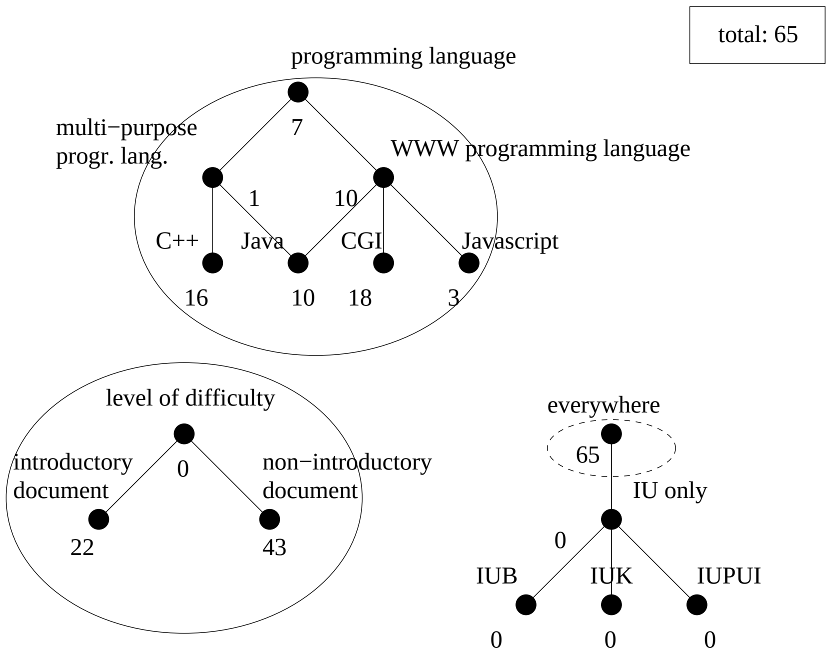

Fig. 7. The query "'programming language' (incl) AND 'everywhere' (excl) AND 'level of difficulty' (incl)"

Acknowledgments

I wish to thank Elin Jacob, Charles Davis, Bernhard Ganter, Jonathan Bolte and the UITS Knowledge Base Team and two anonymous reviewers for hints and discussions of some aspects of this paper.

References

Carpineto, C., & Romano, G. (1995). *Automatic construction of navigable concept networks characterizing text databases.* In M. Gori & G. Soda (Eds.), Topics in Artificial Intelligence. LNAI 992-Springer, pp. 67-78.

Davis, Charles, & McKim, Geoffrey (1999). *Systematic Weighting and Ranking: Cutting the Gordian Knot.* Journal of the American Society for Information Science, 50, 626-628.

Fairthorne, R. A. (1956). *The Patterns of Retrieval.* American Documentation, 7, 65-70.

Godin, R., Missaoui, R., & April, Alain (1993). *Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods.* International Journal of Man-Machine Studies, 38, 747-767.

Ganter, Bernhard, & Wille, Rudolf (1999). *Formal Concept Analysis. Mathematical Foundations.* Berlin-Heidelberg-New York: Springer.

Groh, B., Strahringer, S., & Wille, R. (1998). *TOSCANA-Systems Based on Thesauri*. In M. L. Mugnier, & M. Chein (Eds.), Conceptual structures: theory, tools and applications. LNAI 1453. Springer, pp. 127-138.

UITS (1999). *Indiana University Knowledge Base [On-line]*. Available: http://kb.indiana.edu/info/infopage.html.

Kollewe, W.; Sander, C.; Schmiede, R., & Wille, R. (1995). *TOSCANA als Instrument der bibliothekarischen Sacherschließung.* In H. Havekost, & H.-J. Wätjen (Eds.), Aufbau und Erschließung begrifflicher Datenbanken. Oldenburg: BIS-Verlag, pp. 95-114.

Lengnink, K. (2000). *Ähnlichkeit als Distanz in Begriffsverbänden.* In G. Stumme, & R. Wille (Eds.), Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Berlin-Heidelberg: Springer.

Mooers, C. N. (1958). *A mathematical theory of the use of language symbols in retrieval.* In Proc. Int. Conf. Scientific Information. Washington D.C.

Pedersen, Gert Schmeltz (1993). *A Browser for Bibliographic Information Retrieval on an Application of Lattice Theory.* ACM-SIGIR'93, Pittsburgh, PA, pp. 270-279.

Priss, Uta (1997). *A Graphical Interface for Document Retrieval Based on Formal Concept Analysis.* In E. Santos (Ed.), Proceedings of the 8th Midwest Artificial Intelligence and Cognitive Science Conference. AAAI Technical Report CF-97-01.

Priss, Uta, & Jacob, Elin (1998). *A Graphical Interface for Faceted Thesaurus Design.* In E. Jacob (Ed.), Proceedings of the 9th ASIS SIG/CR Classification Research Workshop, pp. 107-118.

Priss, Uta, & Jacob, Elin (1999). *Utilizing Faceted Structures for Information Systems Design.* Proceedings of the 62st Annual Meeting of ASIS, 203-212.

Rock, T., & Wille, R. (2000). *Ein TOSCANA-Erkundungssystem zur Literatursuche.* In G. Stumme, & R. Wille (Eds.), Begriffliche Wissensverarbeitung. Methoden und Anwendungen. Berlin-Heidelberg: Springer.

Salton, Gerard (1968). *Automatic Information Organization and Retrieval.* McGraw-Hill, New York.

Skorsky, Martin (1997). *Graphische Darstellung eines Thesaurus*. Deutscher Dokumentartag, Regensburg.

Sloane, N. J. A. (1999). *On-Line Encyclopedia of Integer Sequences [On-line].* Available:
http://akpublic.research.att.com/∼njas/sequences/index.html

Soergel, Dagobert (1967). *Mathematical Analysis of Documentation Systems.* Information Storage and Retrieval, 3, pp. 129-173.