Conceptual Clustering with Iceberg Concept Lattices

Gerd Stumme,* **Rafik Taouil,**** **Yves Bastide,***** **Lotfi Lakhal**†

*Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), Universität Karlsruhe (TH), D–76128 Karlsruhe, Germany; stumme@aifb.unikarlsruhe.de

**INRIA Lorraine, LORIA, BP 239, F–54506 Vandoeuvre–lès–Nancy, France; rafik.taouil@loria.fr

***Laboratoire d'Informatique (LIMOS), Université Blaise Pascal, Complexe Scientifique des Cézeaux, 24 Av. des Landais, F–63177 Aubière Cedex, France; bastide@libd2.univbpclermont.fr

†LIM, CNRS FRE2246, Université de la Méditerranée, Case 90, 163 Avenue de Luminy, F–13288 Marseille Cedex 9, France; lotfi.lakhal@lim.univmrs.fr

Abstract. We introduce the notion of iceberg concept lattices and show their use in Knowledge Discovery in Databases (KDD). Iceberg lattices are a conceptual clustering method, which is well suited for analyzing very large databases. They also serve as a condensed representation of frequent itemsets, as starting point for computing bases of association rules, and as a visualization method for association rules. Iceberg concept lattices are based on the theory of Formal Concept Analysis, a mathematical theory with applications in data analysis, information retrieval, and knowledge discovery.

Keywords. Formal Concept Analysis, Conceptual Clustering, Knowledge Discovery, Visualization, Lattices

1 Introduction

Concept Lattices are used to represent conceptual hierarchies which are inherent in data. They are the core of the mathematical theory of Formal Concept Analysis (FCA). Introduced in the early 1980ies as a formalization of the concept of 'concept' [37], FCA has over the years grown to a powerful theory for data analysis, information retrieval, and knowledge discovery [32]. In Artificial Intelligence (AI), FCA is used as a knowledge representation mechanism [39] and as conceptual clustering method [38, 7, 23]. In database theory, FCA has been extensively used for class hierarchy design and management [24, 40, 9, 36, 28, 11]. Its usefulness for the analysis of data stored in relational databases has been demonstrated with the commercially used management system TOSCANA for Conceptual Information Systems [35].

A current research domain common to both the AI and the database community is Knowledge Discovery in Databases (KDD). Here FCA has been used as a method for conceptual clustering [38, 7, 12, 35, 23], a formal framework for implication and association rules discovery and reduction [19, 26, 4, 31], and for improving the response times of algorithms for mining association rules [25, 26, 5]. The interaction of FCA and KDD in general has been discussed in [33] and [14].

In this paper we present a new approach of conceptual clustering with FCA: iceberg concept lattices. Iceberg concept lattices show only the topmost part of a concept lattice. The extensions of the concepts provide the clusters, and the intensions their descriptions. Beside conceptual clustering, iceberg concept lattices have different uses in KDD: as a visualization method — especially for very large databases —, as a condensed representation of frequent itemsets, as a base of association rules, and as a visualization tool for association rules.

In [30], we have presented the algorithm TITANIC, as a new, efficient algorithm for computing concept lattices. As this algorithm is based on a levelwise approach [2, 22] which is in line with our pruning criterion (the support of a concept), it can easily be adapted to compute iceberg concept lattices.[1]

In the next section, we recall the basics of FCA. In Section 3, we introduce iceberg concept lattices and explain their use as conceptual clustering method by an example. Section 4 lists some typical applications. Section 5 concludes the article.

[1] One just has to add "…and $X : s \geq minsupp$" in Line 8 of Algorithm 1 in [30].

2 Formal Concept Analysis

Since concepts are necessary for expressing human knowledge, any knowledge management process benefits from a comprehensive formalization of concepts. FCA offers such a formalization by mathematizing the concept of ‘concept’ as a unit of thought constituted of two parts: its extension and its intension [37, 10]. This understanding of ‘concept’ is first mentioned explicitly in the Logic of Port Royal [3] and has been established in the German standard DIN 2330 and the International Standard ISO 704.

We recall the basics of Formal Concept Analysis as far as they are needed for this paper. A more extensive overview is given in [10].

To allow a mathematical description of extensions and intensions, FCA starts with a (formal) context.

Definition 2.1 *A* formal context *is a triple* $\mathbb{K} := (G, M, I)$ *where* G *and* M *are sets and* $I \subseteq G \times M$ *is a binary relation. The elements of* G *are called* objects *and the elements of* M attributes*. The inclusion* $(g, m) \in I$ *is read “object* g *has attribute* m*”. For* $A \subseteq G$*, we define* $A' := \{m \in M \mid \forall g \in A\colon (g, m) \in I\}$*; and for* $B \subseteq M$*, we define dually* $B' := \{g \in G \mid \forall m \in B\colon (g, m) \in I\}$*.*

We assume — in this article — that all sets are finite, especially G and M.

Definition 2.2 *A* formal concept *is a pair* (A, B) *with* $A \subseteq G$*,* $B \subseteq M$*,* $A' = B$ *and* $B' = A$*. (This is equivalent to* $A \subseteq G$ *and* $B \subseteq M$ *being maximal with* $A \times B \subseteq I$*.)* A *is called* extent *and* B *is called* intent *of the concept.*

The set $\mathfrak{B}(\mathbb{K})$ *of all concepts of a formal context* $\mathbb{K}$ *together with the partial order* $(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2$ *(which is equivalent to* $B_1 \supseteq B_2$*) is called* concept lattice *of* $\mathbb{K}$*.*

Example: As running example, we use the MUSHROOM database from the *UCI KDD Archive* (`http://kdd.ics.uci.edu/`). It consists of a database with 8,416 objects (mushrooms) and 22 (nominally valued) attributes. We obtain a formal context by creating one (Boolean) attribute for each of the 80 possible values of the 22 database attributes. The resulting formal context has thus 8,416 objects and 80 attributes. In order to explain FCA by a small example, we restrict ourselves first to a very limited sub-context, namely the first ten objects, and 13 attributes. This restricted formal context is shown in Figure 1. A line diagram of its concept lattice is shown in Figure 2.

	edible	poisonous	cap shape: convex	cap shape: flat	cap surface: fibrous	cap surface: scaly	cap surface: smooth	cap color: brown	cap color: buff	cap color: gray	cap color: red	cap color: white	cap color: yellow
Mushroom 1	×		×				×					×	
Mushroom 2	×		×		×					×			
Mushroom 3	×			×	×							×	
Mushroom 4	×			×	×			×					
Mushroom 5	×		×		×					×			
Mushroom 6	×		×			×					×		
Mushroom 7		×		×	×					×			
Mushroom 8		×		×	×					×			
Mushroom 9		×		×	×								×
Mushroom 10		×		×			×		×				

Figure 1 Formal context about mushrooms

In the *line diagram*, the name of an object g is always attached to the circle representing the smallest concept with g in its extent; dually, the name of an attribute m is always attached to the circle representing the largest concept with m in its intent. This allows us to read the context relation from the diagram because an object g has an attribute m if and only if there is an ascending path from the circle labeled by g to the circle labeled by m. The extent of a concept consists of all objects whose labels are below in the hierarchy, and the intent consists of all attributes attached to concepts above in the hierarchy. For example, the concept without label in the middle of the diagram has {Mushroom 2, Mushroom 5, Mushroom 4} as extent, and {edible, cap surface: fibrous, cap shape: flat} as intent.

For $X, Y \subseteq M$, we say that the *implication* $X \implies Y$ *holds* in the context, if each object having all attributes in X also has all attributes in Y (i. e., an implication is an association rule[2] with 100% confidence). For instance, the implication {cap shape: flat, cap surface: smooth} $\implies$ {cap color: buff, poisonous} holds in the context. (Of course it may not hold any longer when we enlarge the set of objects under consideration.)

Implications can be read directly in the line diagram: the largest concept having both ‘cap shape: flat’ and ‘cap surface: smooth’ in its intent is just the concept labeled by ‘cap color: buff’ — which on its turn lies below the concept labeled by ‘poisonous’. In the next section is discussed how also the association rules with less than 100 % confidence can by visualized in the line diagram.

Beside association rule mining, FCA has been applied in a wide range of application domains, including medicine, psychology, social sciences, linguistics, information sciences, machine and civil engineering etc. (cf. [32]). Over all, FCA has been used in more

[2]An association rule is a pair $X \to Y$ with $X, Y \subseteq M$. Its *support* is defined by $\mathrm{supp}(X \to Y) := |(X \cup Y)'|/|G|$, and its *confidence* by $\mathrm{conf}(X \to Y) := |(X \cup Y)'|/|X'|$. See [1].

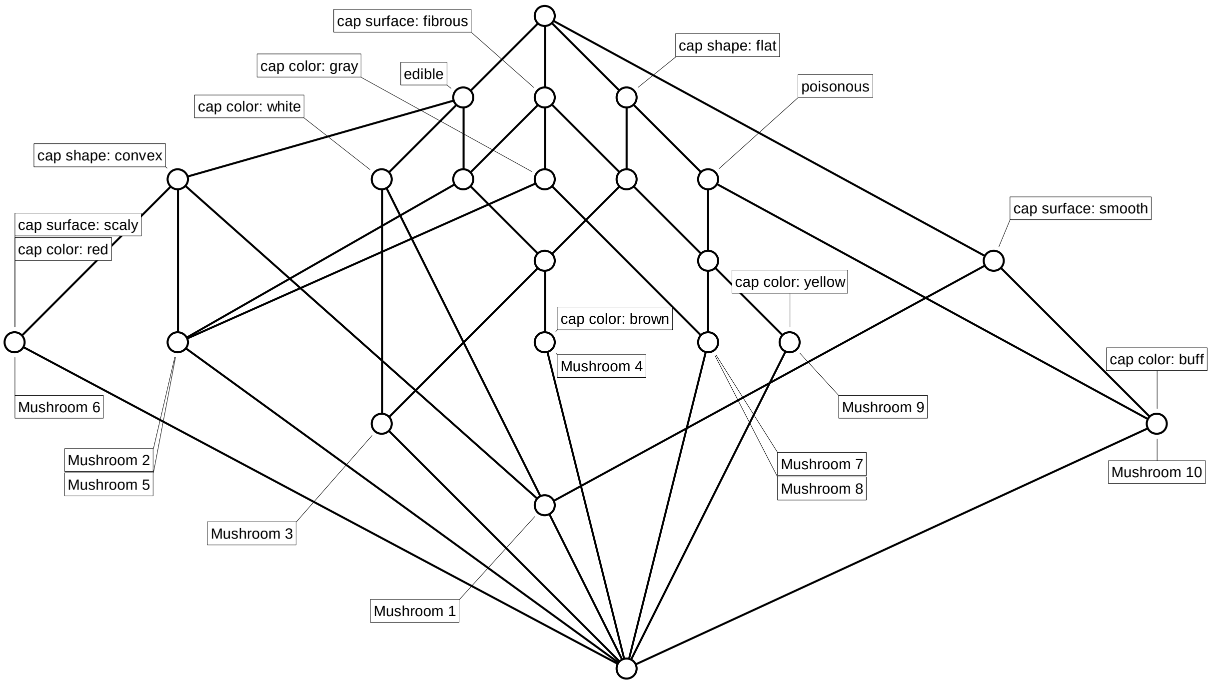

Figure 2 The concept lattice of the context in Figure 1

than 200 projects, both on the scientific and the commercial level. For instance, FCA has been applied for analyzing data of children with diabetes [27], for developing qualitative theories in music esthetics [20], for managing emails [8], for database marketing [14], and for an IT security management system [6].

3 Iceberg Concept Lattices

The previous example was unsatisfying insofar as it was restricted to a very small and — more important — arbitrarily chosen set of objects. On the other hand, this restriction allowed us to display the entire concept lattice. In the worst case, the size of a concept lattices is exponential in the size of the context. Hence for most applications one has to consider strategies (other than arbitrarily reducing the context) for dealing with such large concept lattices. We present an approach based on *frequent itemsets* as known from data mining [1]: Our *iceberg concept lattices* will consist only of the top-most concepts of the concept lattice. These are the concepts which provide the most global structuring of the domain:

Definition 3.1 *Let $B \subseteq M$, and let* $\mathrm{minsupp} \in [0,1]$. *The* support count *of the attribute set (also called itemset) B in $\mathbb{K}$ is* $\mathrm{supp}(B) := |B'|/|G|$. *$B$ is said to be a* frequent *attribute set if* $\mathrm{supp}(B) \geq \mathrm{minsupp}$.

A concept is called frequent concept *if its intent is frequent. The set of all frequent concepts of a context $\mathbb{K}$ is called the* iceberg concept lattice *of the context $\mathbb{K}$.*

Because the support function is monotonously decreasing (i. e., $B_1 \subseteq B_2 \implies \mathrm{supp}(B_1) \geq \mathrm{supp}(B_2)$), the iceberg concept lattice is an order filter of the whole concept lattice, and thus in general only a sup-semi-lattice. But when we add a new bottom element, it becomes a lattice again. This makes it possible to apply the same algorithm (which will be introduced in the following sections) for computing concept lattices and iceberg concept lattices. But before talking about their computation, let's have a closer look to iceberg concept lattices:

Example: Now we consider the whole MUSHROOM database. Its concept lattice consists of 32,086 concepts, hence is by far too large to be displayed. But for a first glance, it is sufficient to see its top-most part: Figure 3 shows the MUSHROOM iceberg concept lattice for a minimum support of 85 %.

In the diagram one can clearly see that all mushrooms in the database have the attribute 'veil type: partial'. Furthermore the diagram tells us that the three next-most attributes are: 'veil color: white' (with 97.62 % support), 'gill attachment: free' (97.43 %), and 'ring number: one' (92.30 %). There is no other attribute having a support higher than 85 %. But even the combination of all these four concepts is frequent (with respect to our threshold of 85 %): 89.92 % of all mushrooms in our database have one ring, a white partial

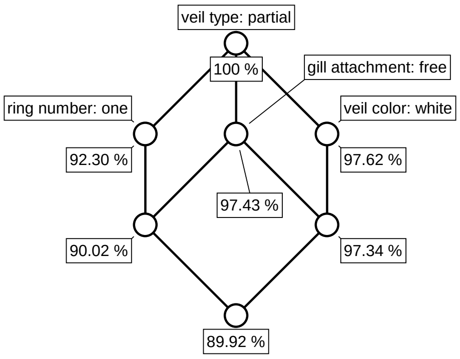

Figure 3 Iceberg concept lattice of the mushroom database with $\mathrm{minsupp} = 85\,\%$

veil, and free gills. This concept with a quite complex description contains more objects than the concept described by the fifth-most attribute, which has a support below our threshold of 85 %, since it is not displayed in the diagram.

In the diagram, we can detect the implication

$$\{\text{ring number: one, veil color: white}\} \Longrightarrow \{\text{gill attachment: free}\}\ .$$

It is indicated by the fact that there is no concept having 'ring number: one' and 'veil color: white' (and 'veil type: partial') in its intent, but not 'gill attachment: free'. This implication has a support of 89.92 % (and as it is an implication, a confidence of 100 %). Unlike the implications in Example 1 (which held for the ten objects under consideration only), this implication is globally valid, i. e., it does not change when we consider a different minimum support.

If we want to see more details, we have to decrease the minimum support. Figure 4 shows the MUSHROOM iceberg concept lattice for a minimum support of 70 %. One observes that, of course, its top-most part is just the iceberg lattice for $\mathrm{minsupp} = 85\,\%$. Additionally, we obtain five new concepts, having the possible combinations of the next-most attribute 'gill spacing: close' (having support 81.08 %) with the previous four attributes. The fact that the combination {gill spacing: close, veil type: partial, gill attachment: free} is not realized as a concept intent indicates another implication:

$$\{\text{gill attachment: free, gill spacing: close}\} \Longrightarrow \{\text{veil color: white}\} \qquad (*)$$

This implication has 78.52 % support (the support of the most general concept having all three attributes in its intent) and — being an implication — 100 % confidence.

By further decreasing the minimum support, we discover more and more details. Figure 5 shows the MUSHROOM iceberg concept lattice for a minimum support of 55 %. It shows four more partial copies of the 85 % iceberg lattice, and three new, single concepts.

The observation that the top-most part of the iceberg lattice appears partially again in combination with other attributes can be used for an alternative visualization: Figure 6 shows the iceberg concept lattice as a *nested line diagram*. The diagram provides exactly the same information as Figure 5, but in a more structured way.

Each of the 'satellites' contains a partial copy of the top-most iceberg lattice. Only those concepts are copied which are, together with the new attribute(s), still frequent. The lines of the outer diagram have to be read as a bundle of parallel lines, linking corresponding concepts. For instance, the concept on the right side of the diagram labeled by '78.80 %' is not only an immediate subconcept of the one labeled by '81.08 %, but also of the one labeled by '97.62 %'.

The empty circles indicate *unrealized concepts*: They are still frequent, but all objects in an unrealized concept share at least one more attribute. For instance, the unrealized concept on the right side left of the concept labeled by '78.80 %' has as intent {gill spacing: close, gill attachment: free, veil type: partial}. But implication (*) tells us that all objects having these attributes also have the attribute 'veil color: white'. Therefore, 'veil color: white' has to be in each realized concept which contains the three other attributes. The largest of them is just the first realized concept below: the one with 78.52 % support. This way, each unrealized concept indicates an implication: the attributes of its intent always imply all attributes in the intent of its largest realized subconcept. For instance, the two unrealized concepts below the attribute 'no bruises' indicate the implications

$$\{\text{no bruises, gill attachment: free}\} \Longrightarrow \{\text{veil color: white}\}$$

$$\{\text{no bruises, veil color: white}\} \Longrightarrow \{\text{gill attachment: free}\}$$

respectively, each having 57.22 % support.

For attributes which are labeled at concepts having no subconcepts in the diagram, we cannot decide whether they are part of interesting implications. For instance, the diagram does not show whether there is an implication having 'stalk color below ring: white' in its premise or conclusion (other than the trivial implication $\{\text{stalk color below ring: white}\} \Longrightarrow \{\text{veil type: partial}\}$). If there are any such rules, then their support

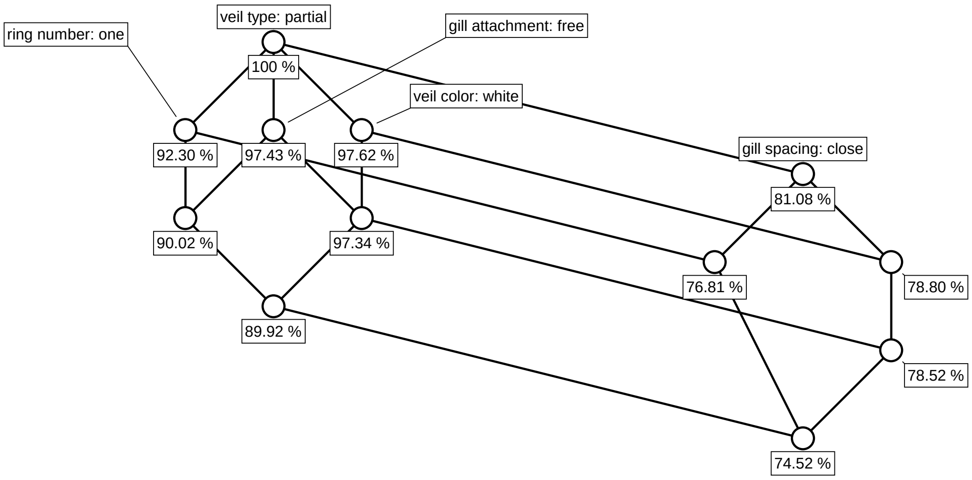

Figure 4 Iceberg concept lattice of the mushroom database with $\mathrm{minsupp} = 70\,\%$

is below the actual minimum support of 55 %. In order to study them, the threshold has to be decreased further.

In the way nested line diagrams are introduced in [38], the attributes are grouped manually according to their semantics. Related attributes are grouped together. This usually involves a human expert to decide which attributes are related. The support function, on the other hand, allows an automatic grouping: In Figure 6, the inner diagram contains the top-most attributes, the outer diagram the next-most attributes. The resulting diagram shows the most important attributes for structuring the domain. The knowledge engineer only has to fix the minimum support thresholds for the different layers.

Observe that the iceberg concept lattices in this example are used for *conceptual clustering*, or *unsupervised learning*. Our aim was to gain new insights about the mushrooms in the database, independent from a specific purpose. In particular, the aim was not to learn how to distinguish between poisonous and edible mushrooms. The question if and how iceberg concept lattices can be used in such a *supervised learning* scenario is an interesting open problem.

Up to now, we have discussed the use of iceberg concept lattices as a conceptual clustering technique, equipped with a visualization method, which is very well suited especially for analyzing *very large* databases containing strongly correlated data. Now we briefly discuss some more uses of iceberg concept lattices in KDD:

A condensed representation of frequent itemsets. The computation of frequent attribute sets [itemsets] is the first (and most expensive) step in the computation of association rules. One reason is that one needs to count the support for each itemset. By using the fact that $\mathrm{supp}(B) = \mathrm{supp}(B'')$, for $B \subseteq M$, we can derive the supports of all itemsets from the supports of the frequent concept intents only. In strongly correlated data, only relatively few of the frequent itemsets are also concept intents. Hence only few support counts have to be effected in the database.

A starting point for computing bases of association rules. One problem in mining association rules is the large number of rules which are usually returned. In [4], different bases for association rules are introduced, which prune redundant rules, but from which all valid rules can still be derived. The computation of the bases does not require all frequent itemsets, but only frequent concept intents.

A visualizing technique for association rules. We have already discussed how implications (i. e., association rules with 100 % confidence) can be read from the line diagram. The Luxenburger basis for approximate association rules (i. e., association rules with less than 100 % confidence), which is presented in [34], can also be visualized directly in the line diagram of an iceberg concept lattice. The Luxenburger basis is derived from [19]. It contains only those rules $B_1 \to B_2$ where B_1 and B_2 are frequent concept intents, and the concept (B_1', B_1) is an immediate subconcept of (B_2', B_2). Hence there corresponds to each approximate rule in the Luxenburger base exactly one edge in the line diagram. Figure 7 visualizes all rules in

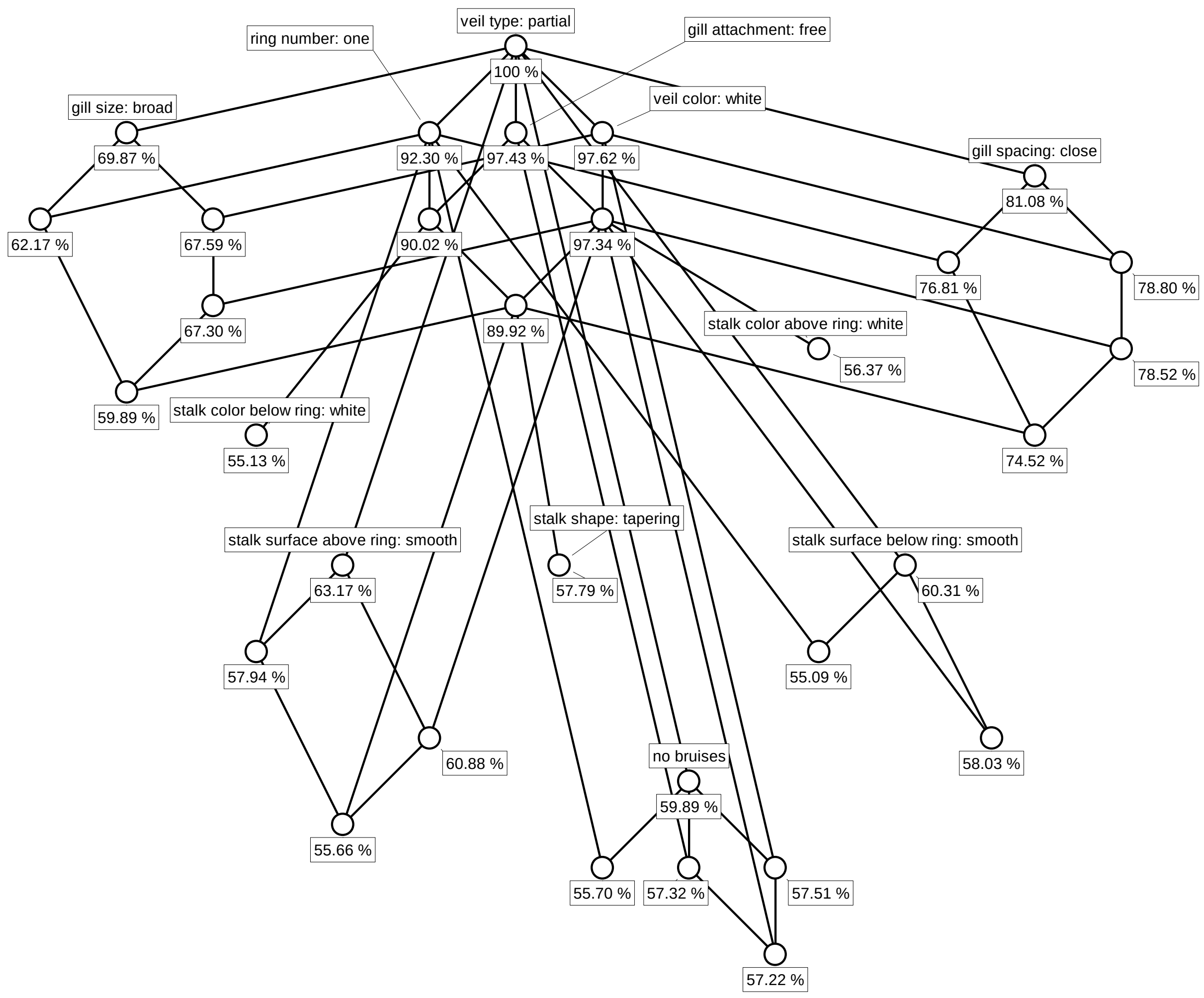

Figure 5 Iceberg concept lattice of the mushroom database with $\mathrm{minsupp} = 55\,\%$

the Luxenburger basis for minsupp = 70 % and minconf = 95 %. For instance, the rightmost arrow stands for the association rule {veil color: white, gill spacing: close} → {gill attachment: free}, which holds with a confidence of 99.6 %. Its support is the support of the concept the arrow is pointing to: 78.52 %, as shown in Figure 4. Edges without label indicate that the confidence of the rule is below the minimum confidence threshold.

4 Some Typical Applications

In Section 3, we have already discussed the use of (iceberg) concept lattices for knowledge discovery and conceptual clustering. Here we give another, real-world example of a KDD application:

Database marketing. The purpose of database marketing is the study of customers and their buying behavior in order to create and validate marketing strategies. In [14], the use of iceberg concept lattices for database marketing in a Swiss department store is discussed in more detail. In that scenario, the object set G consists of all customers of the warehouse paying by credit card, and the attribute set M consists of attributes describing the customers (e. g., 'lives in Western Switzerland') and their buying behavior (e. g., 'has spent more than 1000 Swiss francs in the last year'). For a given set X of attributes, the weight function returns the number of customers fulfilling all attributes in X. By decreasing the minimum support, one can study the customer behavior in more and more detail.

The use of (iceberg) concept lattices is not only restricted to knowledge discovery. Here we give some more examples of typical applications, in which TITANIC can be applied:

Configuration space analysis. In software re-engineering, one task is to analyze the source code of a given program where no (or relatively few)

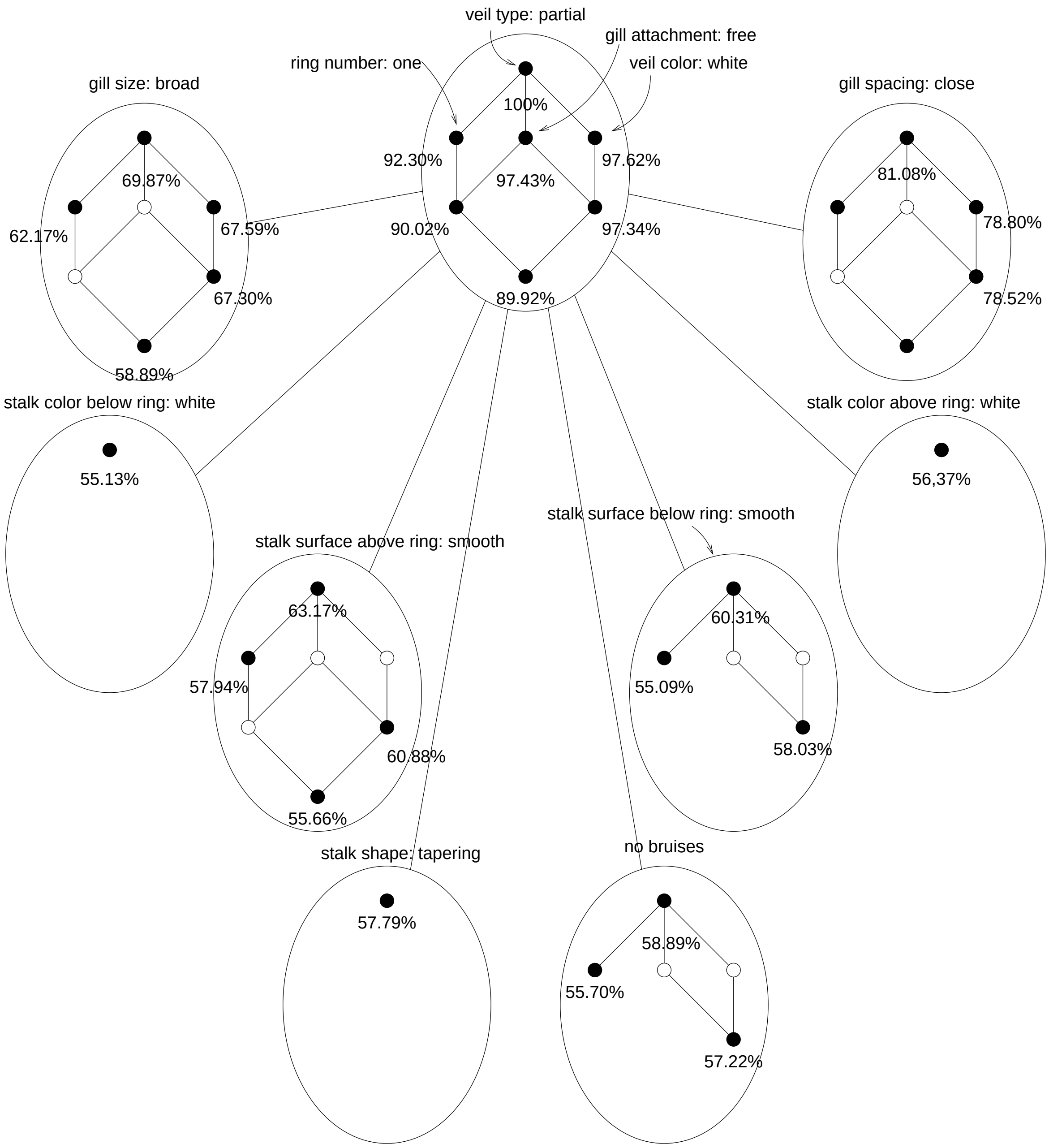

Figure 6 Nested line diagram of the iceberg concept lattice in Figure 5

documentation is given. In [17], the use of Formal Concept Analysis for analyzing the configuration space of C++ programs is discussed. In the described scenario, iceberg concept lattices could be introduced quite naturally. The set G of objects contains the lines of code, the set M consists basically of the C++ preprocessor symbols which appear in the code, and the relation I indicates which lines of code are governed by which preprocessor symbols. Instead of computing the whole concept lattice, one can restrict the computation to the top-level groupings of code pieces by using TITANIC. The weight function returns, for a set X of preprocessor symbols, the number of lines of code which are governed by all preprocessor symbols in X.

Transformation of class hierarchies. In object-oriented languages, one aim is to simplify the class hierarchy according to a (number of) given program(s). In [29], this problem has been attacked by using concept lattices. In the scenario, the set M of attributes contains all data members and methods of a given class hierarchy, and the set G of objects consists of all variables and pointers of the program(s). The relation I basically indicates which variables and pointers are related to which data members and methods.The re-

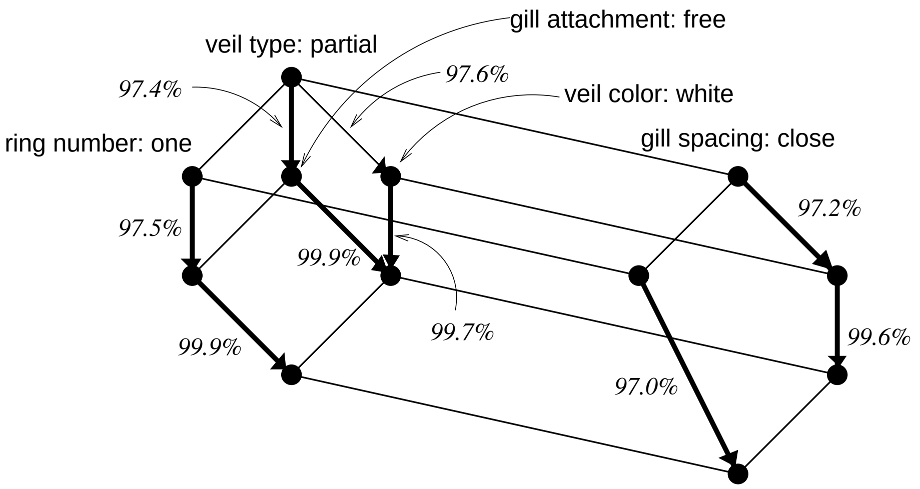

Figure 7 Visualization of the Luxenburger basis for $\mathrm{minsupp} = 70\,\%$ and minconf$= 95\,\%$

sulting concept lattice provides an improved hierarchy which can be used for restructuring the class hierarchy according to software engineering principles without the need to modify the source code. The computation of the concept lattice can be done by using as weight function the function which returns, for a given set X of data members and methods, the number of variables and pointers related to all elements in X.

Ontology Learning. Ontologies are "explicit specification[s] of a conceptualization" [13]. They usually consist of a set of concepts (not to be confused with formal concepts from FCA), a hierarchical is-a relation and other (non-hierarchical) relations between the concepts, and eventually axioms describing constraints on the relations and concepts. One task in learning ontologies from data is the construction of the is-a hierarchy. Suppose that the concepts are already learned (e. g., by applying linguistic and statistical methods [21]) and stored in the set M. The set G contains tuples of a relational database, or documents annotated with the concepts. The relation I indicates if a tuple includes a concept, or if a document is annotated with a concept. TITANIC uses the weight function, which assigns to a set X of ontology concepts the number of documents/tuples related to all concepts in X. The resulting iceberg concept lattice provides an is-a hierarchy on the set of the ontology concepts. Additionally, it suggests new concepts which may simplify the structure of the concept hierarchy.

Another situation where a weight function arises naturally in the computation of a closure system is the following. This scenario is more difficult to state in terms of a formal context:

Discovery of functional dependencies. One important task of logical database tuning is the discovery of minimal functional dependencies from database relations [15, 18]. This is equivalent to computing a closure system on the set M of all database attributes. The closed sets are just those which are closed under all functional dependencies which hold in the database. TITANIC can be applied for this computation, using as weight of a given attribute set X the minimal number of rows which have to be deleted from the database such that X is closed under all functional dependencies which are valid for the remaining rows. This weight function is derived from the g_3 measure introduced in [16]. For this application, all 'min' in this paper have to be replaced by 'max' (refer to Remark 2).

5 Conclusion

The paper shows the use of iceberg concept lattices as a conceptual clustering method, a condensed representation of frequent itemsets, and an efficient visualization technique for conceptual hierarchies derived from very large databases. Typical examples for its application are listed in the paper.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. SIGMOD Conf.*, pages 207–216, 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB Conf.*, pages 478–499, 1994. (Expanded version in IBM Report RJ9839).
3. A. Arnauld and P. Nicole. *La logique ou l'art de penser — contenant, outre les règles communes, plusieurs observations nouvelles, propres à former le jugement.* Ch. Saveux, Paris, 1668.
4. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal nonredundant association rules using frequent

closed itemsets. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, editors, *Computational Logic — CL. Proc. 1st Intl. Conf. on CL (6th Intl. Conf. on Database Systems)*, number 1861 in LNAI, pages 972–986, Heidelberg, 2000.

5. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *Sigkdd Explorations*, 2(2):71–80, 2000. Special Issue on Scalable Algorithms.
6. K. Becker, G. Stumme, R. Wille, U. Wille, and M. Zickwolff. Conceptual information systems discussed through an itsecurity tool. In R. Dieng and O. Corby, editors, *Knowledge Engineering and Knowledge Management. Methods, Models, and Tools. Proc. EKAW '00*, number 1937 in LNAI, pages 352–365, Heidelberg, 2000. Springer.
7. C. Carpineto and G. Romano. GALOIS: An ordertheoretic approach to conceptual clustering. In *Proc. ICML 1993*, pages 33–40. Morgan Kaufmann Prublishers, 1993.
8. R. Cole and G. Stumme. Cem – a conceptual email manager. In B. Ganter and G. W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues. Proc. ICCS '00*, LNAI, pages 438–452, Heidelberg, 2000. Springer.
9. H. Dicky, C. Dony, M. Huchard, and Th. Libourel. On automatic class insertion with overloading. In *OOPSLA*, pages 251–267, 1996.
10. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Heidelberg, 1999.
11. R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, and T. Chau. Design of class hierarchies based on concept (Galois) lattices. *TAPOS*, 4(2):117–134, 1998.
12. R. Godin and R. Missaoui. An incremental concept formation approach for learning from databases. *TCS*, 133(2):387–419, 1994.
13. T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *Intl. J. of Human and Computer Studies*, 46((2/3)):293–310, 1997.
14. J. Hereth, G. Stumme, U. Wille, and R. Wille. Conceptual knowledge discovery and data analysis. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Structures. Proc. ICCS 2000*, number 1867 in LNAI, pages 421–437, Heidelberg, 2000. Springer.
15. Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: an efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.
16. J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *TCS*, 149(1), 1995.
17. M. Krone and G. Snelting. On the inference of configuration structures from source code. In *Proc. 16th Intl. Conference on Software Engineering*, pages 49–57. IEEE Comp. Soc. Press, May 1994.
18. S. Lopes, J.-M. Petit, and L. Lakhal. Efficient discovery of functional dependencies and amstrong relations. In *Proc. EDBT 2000*, number 1777 in LNCS, pages 350–364, Heidelberg, 2000. Springer.
19. M. Luxenburger. Implications partielles dans un contexte. *Mathématiques, Informatique et Sciences Humaines*, 29(113):35–55, 1991.
20. K. Mackensen and U. Wille. Qualitative text analysis supported by conceptual data systems. *Quality and Quantity: Internatinal Journal of Methodology*, 2(33):135–156, 1999.
21. A. Mädche and S. Staab. Mining ontologies from text. In R. andO Corby Dieng, editor, *Knowledge Engineering and Knowledge Management. Methods, Models, and Tools. Proc. EKAW '00*, number 1937 in LNAI, pages 189–202, Heidelberg, 2000. Springer.
22. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
23. G. Mineau, G. and and R. Godin. Automatic structuring of knowledge bases by conceptual clustering. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):824–829, 1995.
24. M. Missikoff and M. Scholl. An algorithm for insertion into a lattice: application to type classification. In *Proc. 3rd Intl. Conf. FODO 1989*, number 367 in LNCS, pages 64–82, Heidelberg, 1989. Springer.
25. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT '99*, number 1540 in LNCS, pages 398–416, Heidelberg, 1999. Springer.
26. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Journal of Information Systems*, 24(1):25–46, 1999.
27. P. Scheich, M. Skorsky, F. Vogt, C. Wachter, and R. Wille. Conceptual data systems. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification*, pages 72–84. Springer, BerlinHeidelberg, 1993.
28. I. Schmitt and G. Saake. Merging inheritance hierarchies for database integration. In *Proc. 3rd IFCIS Intl. Conf. on Cooperative Information Systems*, pages 122–131, New York City, Nework, USA, August2022 1998.
29. G. Snelting and F. Tip. Reengineering class hierarchies using concept analysis. In *Proc. ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 99–110, November 1998.
30. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Fast computation of concept lattices using data mining techniques. In *Proc. 7th Intl. Workshop on Knowledge Representation Meets Databases*, pages 21–22, Berlin, August 2000. CEURWorkshop Proceeding. `http://sunsite.informatik.rwthaachen.de/Publications/CEURWS/`.
31. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Intelligent structuring and reducing of association rules with formal concept analysis. In *Proc. KI 2001*, LNAI, Heidelberg, 2001. Springer. (to appear).
32. G. Stumme and R. Wille, editors. *Begriffliche Wissensverarbeitung – Methoden und Anwendungen*. Springer, Heidelberg, 2000.
33. G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery in databases using formal concept analysis methods. In J. M. Żytkow and M. Quafofou, editors, *Principles of Data Mining and Knowledge Discovery. Proc. 2nd European Symposium on PKDD '98*, number 1510 in LNAI, pages 450–458, Heidelberg, 1998. Springer.
34. R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Mining bases for assocition rules using closed sets. In *Proc. 16th Intl. Conf. ICDE 2000*, San Diego, CA, US, February 2000.
35. F. Vogt and R. Wille. TOSCANA – a graphical tool for analyzing and exploring data. In *Graph Drawing 94*, number 894 in LNCS, pages 226–233, Heidelberg, 1995. Springer.
36. K. Waiyamai, R. Taouil, and L. Lakhal. Towards an object database approach for managing concept lattices. In *Proc. 16th Intl. Conf. on Conceptual Modeling*, number 1331 in LNCS, pages 299–312, Heidelberg, 1997. Springer.
37. R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. Reidel, Dordrecht–Boston, 1982.
38. R. Wille. Line diagrams of hierarchical concept systems. *Int. Classif.*, 11(2):77–86, 1984.
39. R. Wille. Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications*, 23(6-9):493–515, 1992.
40. A. Yahia, L. Lakhal, J. P. Bordat, and R. Cicchetti. iO2: An algorithmic method for building inheritance graphs in object database design. In *Proc. 15th Intl. Conf. on Conceptual Modeling*, number 1157 in LNCS, pages 422–437, Heidelberg, 1996. Springer.