Explaining Data with Formal Concept Analysis

Bernhard Ganter$^{1[0000-0003-0767-1379]}$, Sebastian Rudolph$^{1[0000-0002-1609-2080]}$, and Gerd Stumme$^{2[0000-0002-0570-7908]}$

^{1} TU Dresden `firstname.lastname@tu-dresden.de`
^{2} Uni Kassel `stumme@cs.uni-kassel.de`

Abstract. We give a brief introduction into Formal Concept Analysis, an approach to explaining data by means of lattice theory.

Keywords: Formal Concept Analysis · Data Visualization · Attribute Logic.

1 Introduction

Formal Concept Analysis (FCA) is a mathematical discipline which attempts to formalize aspects of human conceptual thinking. For cognitive reasons, humans tend to form categories for objects and situations they encounter in the real world. These groups, defined based on commonalities between their elements, can then be given a name, referred to, and reasoned about in their entirety. They can be ordered by the level of generality or specificity giving rise to what is called “conceptual hierarchies” or “taxonomies”. FCA provides a very simplified, yet powerful and elegant formalization of the notion of “concept” by means of lattice theory.

Over the last four decades, FCA has developed in a versatile scientific field, yielding novel approaches to data visualization and data mining. It greatly contributed to the development of data science and can be seen as a bottom-up approach to explain data by means of hierarchical clustering techniques.

Here, we provide a gentle introduction into the basics of FCA. Thereby, we will omit mathematical proofs of the presented theorems and lemmas; the interested reader may consult [4] for more details.

2 TL;DR – Formal Concept Analysis in a Nutshell

This section is meant to be an ‘appetizer’. It provides a brief overview over Formal Concept Analysis, in order to allow for a better understanding of the overall picture. To this end, this section introduces the most basic notions of Formal Concept Analysis, namely formal contexts, formal concepts, and concept lattices. These definitions will be repeated and discussed in more detail later on.

Formal Concept Analysis (FCA) was introduced as a mathematical theory modeling the notion of ‘concepts’ in terms of lattice theory. To come up with a formal description of concepts and their constituents, extensions and intensions, FCA starts by defining *(formal) contexts.*

Definition 1 A *(formal) context* is a triple $\mathbb{K} := (G, M, I)$, where G is a set whose elements are called *objects*, M is a set whose elements are called *attributes*, and I is a binary relation between G and M (i. e., $I \subseteq G \times M$), where $(g, m) \in I$ is read "object g *has* attribute m". $\Diamond$

This definition captures the basic and immediately graspable idea of a collection of entities, each of which might or might not have certain properties. At the same time, this notion is generic enough to be applicable to a vast variety of situations.

On another note, the interested reader might notice that formal contexts are closely related to *bipartite graphs* (where both objects and attributes are nodes in the graph and edges are connecting each object with its attributes). This link enables the study of bipartite graphs using FCA and, likewise, FCA can profit from known results developed for bipartite graphs.

Figure 1 shows a formal context where the object set G comprises all airlines of the Star Alliance group and the attribute set M lists their destinations.[3] The binary relation I is given by the cross table and describes which destinations are served by which Star Alliance member.

Definition 2 For an object set $A \subseteq G$, let

$$A' := \{m \in M \mid \forall g \in A\colon (g, m) \in I\}$$

and, for an attribute set $B \subseteq M$, let

$$B' := \{g \in G \mid \forall m \in B\colon (g, m) \in I\} \ .$$

A *(formal) concept* of a formal context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The sets A and B are called the *extent* and the *intent* of the formal concept (A, B), respectively. The *subconcept–superconcept relation* $\leq$ is formalized by

$$(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2 \quad (\Longleftrightarrow B_1 \supseteq B_2) \ .$$

The set of all formal concepts of a formal context $\mathbb{K}$ together with the order relation $\leq$ always constitutes a complete lattice,[4] called the *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$. $\Diamond$

Figure 2 visualizes the concept lattice of the context in Figure 1 by means of a *line diagram*. In a line diagram, each node represents a formal concept. A concept $\mathfrak{c}_1$ is a subconcept of a concept $\mathfrak{c}_2$ if and only if there is a path of descending edges from the node representing $\mathfrak{c}_2$ to the node representing $\mathfrak{c}_1$. The name of an object g is always attached to the node representing the smallest concept with g in its extent; dually, the name of an attribute m is always attached to the node

[3] Note that the underlying data is somewhat outdated, if not to say antiquated.

[4] I. e., for each subset of concepts, there is always a unique greatest common subconcept and a unique least common superconcept.

	Latin America	Europe	Canada	Asia Pacific	Middle East	Africa	Mexico	Caribbean	United States
Air Canada	×	×	×	×	×		×	×	×
Air New Zealand		×		×					×
All Nippon Airways		×		×					×
Ansett Australia				×					
The Austrian Airlines Group		×	×	×	×	×			×
British Midland		×							
Lufthansa	×	×	×	×	×	×	×		×
Mexicana	×		×				×	×	×
Scandinavian Airlines	×	×		×		×			×
Singapore Airlines		×	×	×	×	×			×
Thai Airways International	×	×		×				×	×
United Airlines	×	×	×	×			×	×	×
VARIG	×	×		×		×	×		×

Fig. 1. A formal context about the destinations of the Star Alliance members

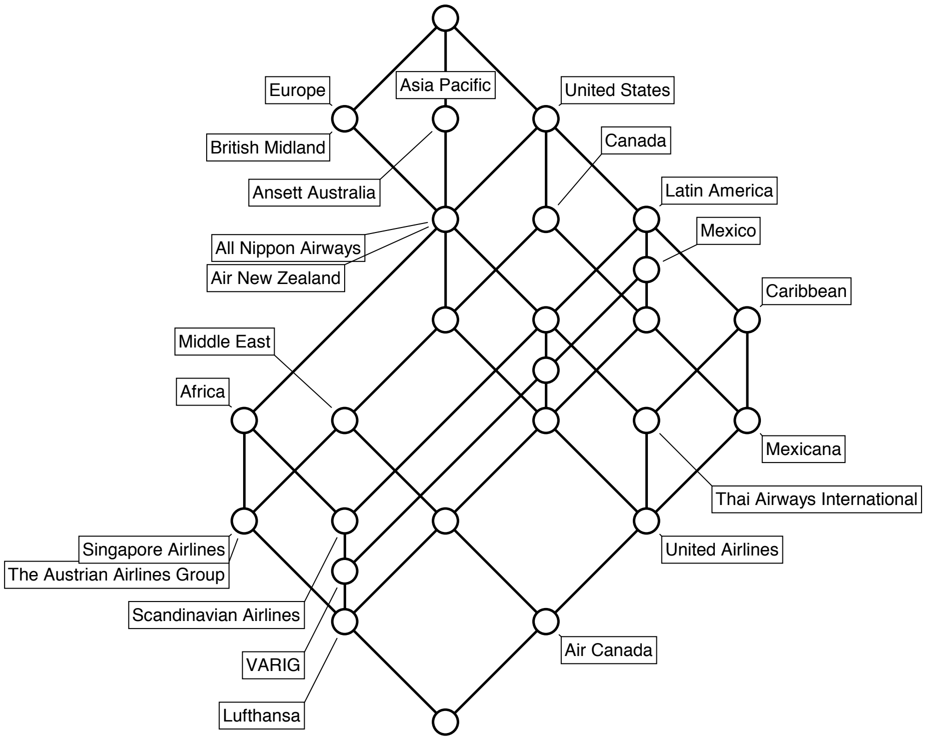


Fig. 2. The concept lattice of the context in Figure 1

representing the largest concept with m in its intent. We can read the context relation from the diagram because an object g has an attribute m if and only if the concept labeled by g is a subconcept of the one labeled by m. The extent of a concept consists of all objects whose labels are attached to subconcepts, and, dually, the intent consists of all attributes attached to superconcepts. For example, the concept labeled by 'Middle East' has {Singapore Airlines, The Austrian Airlines Group, Lufthansa, Air Canada} as extent, and {Middle East, Canada, United States, Europe, Asia Pacific} as intent.

High up in the diagram, we find the destinations which are served by most of the members: Europe, Asia Pacific, and the United States. For instance, besides British Midland and Ansett Australia, all airlines are serving the United States. Those two airlines are located at the top of the diagram, as they serve the fewest destinations — they operate only in Europe and Asia Pacific, respectively.

The further we go down in the concept lattice, the more globally operating are the airlines. The most destinations are served by the airlines close to the bottom of the diagram: Lufthansa (serving all destinations besides the Caribbean) and Air Canada (serving all destinations besides Africa). Also, the further we go down in the lattice, the lesser served are the destinations. For instance, Africa, the Middle East, and the Caribbean are served by relatively few Star Alliance members.

Dependencies between the attributes can be described by implications. For attribute sets $X, Y \subseteq M$, we say that the *implication* $X \to Y$ *holds* in the context, if each object having all attributes in X also has all attributes in Y. For instance, the implication {Europe, United States} $\to$ {Asia Pacific} holds in the Star Alliance context. It can be read directly from the line diagram: the largest concept having both 'Europe' and 'United States' in its intent (i. e., the concept labeled by 'All Nippon Airways' and 'Air New Zealand') also has 'Asia Pacific' in its intent. Similarly, one can detect that the destinations 'Africa' and 'Canada' together imply the destination 'Middle East' (and also 'Europe', 'Asia Pacific', and 'United States').

Concept lattices can also be visualized using *nested line diagrams*. For obtaining a nested line diagram, one splits the set of attributes in two parts, and obtains thus two formal contexts with identical object sets. For each formal context, one computes its concept lattice and a line diagram. The nested line diagram is obtained by enlarging the nodes of the first line diagram and by drawing the second diagram inside. The second lattice is used to further differentiate each of the extents of the concepts of the first lattice. Figure 3 shows a nested line diagram for the Star Alliance context. It is obtained by splitting the attribute set as follows: $M = \{$ Europe, Asia Pacific, Africa, Middle East $\} \cup \{$ United States, Canada, Latin America, Mexico, Caribbean $\}$. The order relation can be read by replacing each of the lines of the large diagram by eight parallel lines linking corresponding nodes in the inner diagrams. The concept lattice given in Figure 2 is embedded (as a join–semilattice) in this diagram, it consists of the solid nodes. The concept mentioned above (labeled by 'Middle East') is for instance represented by the left-most solid node in the lower right part.

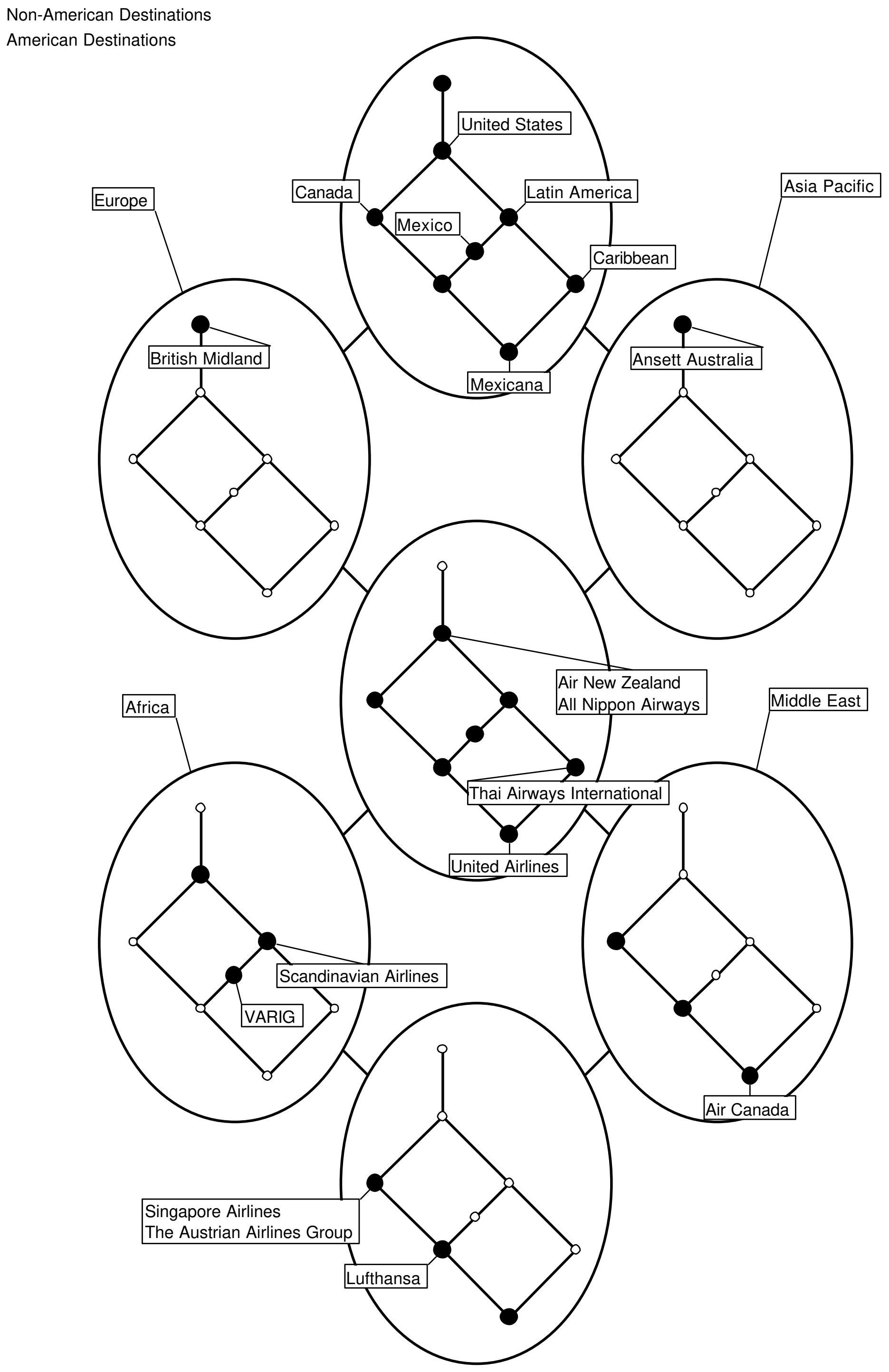


Fig. 3. A nested diagram of the concept lattice in Figure 2

The solid concepts are referred to as 'realized concepts', as, for each of them, the set of all attributes labeled above is an intent of the formal context. The non-realized concepts are not only displayed to indicate the structure of the inner scale, but also because they indicate implications: Each non-realized concept indicates that the attributes in its intent imply the attributes contained in the largest realized concept below. For instance, the first implication discussed above is indicated by the non-realized concept having as intent 'Europe' and 'United States', it is represented by the empty node below the concept labeled by 'British Midland'. The largest realized sub-concept below is the one labeled by 'All Nippon Airways' and 'Air New Zealand' — which additionally has 'Asia Pacific' in its intent. Hence the implication $\{$ Europe, United States $\} \rightarrow \{$ Asia Pacific $\}$ holds. The second implication from above is indicated by the non-realized concept left of the concept labeled by 'Scandinavian Airlines', and the largest realized concept below, which is the one labeled by 'Singapore Airlines' and 'The Austrian Airlines Group'.

This section gave a short introduction to the core notions of FCA. We will discuss most of them (and more advanced topics) in more detail in the remainder of this chapter.

3 Concept Lattices

Formal Concept Analysis studies how *objects* can be hierarchically grouped together according to their common *attributes*. One of the aspects of FCA thus is *attribute logic*, the study of possible attribute combinations. Most of the time, this will be very elementary. Those with a background in Mathematical Logic might say that attribute logic is just Propositional Calculus, and thus Boolean Logic, or even a fragment of this. Historically, the name *Propositional Logic* is misleading: Boole himself used the intuition of attributes ("signs") rather than of propositions. So in fact, attribute logic goes back to Boole.

But our style is different from that of logicians. Our logic is *contextual*, which means that we are interested in the logical structure of concrete data (of the *context*). Of course, the general rules of mathematical logic are important for this and will be utilized.

3.1 Formal Contexts and Cross Tables

Definition 3 A **Formal Context** (G, M, I) consists of two sets G and M and of a binary relation $I \subseteq G \times M$. The elements of G are called the **objects**, those of M the **attributes** of (G, M, I). If $g \in G$ and $m \in M$ are in relation I, we write $(g, m) \in I$ or $g\,I\,m$ and read this as *"object g* **has** *attribute m"*. $\Diamond$

The simplest format for writing down a formal context is a **cross table**: we write a rectangular table with one row for each object and one column for each attribute, having a cross in the intersection of row g with column m iff $(g, m) \in I$. The simplest data type for computer storage is that of a bit matrix.[5]

Note that the definition of a formal context is very general. There are no restrictions about the nature of objects and attributes. We may consider physical objects, or persons, numbers, processes, structures, etc. – virtually everything. Anything that is a *set* in the mathematical sense may be taken as the set of objects or of attributes of some formal context. We may interchange the rôle of objects and attributes: if (G, M, I) is a formal context, then so is the **dual context** (M, G, I^{-1}) (with $(m, g) \in I^{-1} :\iff (g, m) \in I$). It is also not necessary that G and M are disjoint, they need not even be different.

On the other hand, the definition is rather restrictive when applied to real world phenomena. Language phrases like "all human beings" or "all chairs" do not denote sets in our sense. There is no "set of all chairs", because the decision if something is a chair is not a matter of fact but a matter of subjective interpretation. The notion of "formal concept" which we shall base on the definition of "formal context" is much, much narrower than what is commonly understood as a concept of human cognition. The step from "context" to "formal context" is quite an incisive one. It is the step from "real world" to "data". Later on, when we get tired of saying "formal concepts of a formal context", we will sometimes omit the word "formal". But we should keep in mind that it makes a big difference.

3.2 The derivation operators

Given a selection $A \subseteq G$ of objects from a formal context (G, M, I), we may ask which attributes from M are common to all these objects. This defines an operator that produces for every set $A \subseteq G$ of objects the set $A^{\uparrow}$ of their common attributes.

Definition 4 For $A \subseteq G$, we let

$$A^{\uparrow} := \{m \in M \mid g\ I\ m \text{ for all } g \in A\} \ .$$

Dually, we introduce for a set $B \subseteq M$ of attributes

$$B^{\downarrow} := \{g \in G \mid g\ I\ m \text{ for all } m \in B\} \ .$$

These two operators are the **derivation operators** for (G, M, I). $\Diamond$

The set $B^{\downarrow}$ denotes thus the set consisting of those objects in G that have (at least) all the attributes from B.

[5] It is not easy to say which is the *most efficient* data type for formal contexts. This depends, of course, on the operations we want to perform with formal contexts. The most important ones are the derivation operators, to be defined in the next subsection.

Usually, we do not distinguish the derivation operators in writing and use the notation A', B' instead. This is convenient, as long as the distinction is not explicitly needed.

If A is a set of objects, then A' is a set of attributes, to which we can apply the second derivation operator to obtain A'' (more precisely: $(A^{\uparrow})^{\downarrow}$), a set of objects. Dually, starting with a set B of attributes, we may form the set B'', which is again a set of attributes. We have the following simple facts:

Proposition 1 *For subsets $A, A_1, A_2 \subseteq G$ we have*

1. $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$,
2. $A \subseteq A''$,
3. $A' = A'''$.

Dually, for subsets $B, B_1, B_2 \subseteq M$ we have

1'. $B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1'$,
2'. $B \subseteq B''$,
3'. $B' = B'''$.

The reader may confer to [4] for details and proofs. The mathematically interested reader may notice that the derivation operators constitute an (antitone) **Galois connection** between the (power sets of the) sets G and M.

The not so mathematically oriented reader should try to express the statements of the Proposition in common language. We give an example: Statement 1. says that *if a selection of objects is enlarged, then the attributes which are common to all objects of the larger selection are among the common attributes of the smaller selection.* Try to formulate 2. and 3. in a similar manner!

3.3 Formal Concepts, Extent and Intent

In what follows, (G, M, I) always denotes a formal context.

Definition 5 (A, B) is a **formal concept** of (G, M, I) iff

$$A \subseteq G, \quad B \subseteq M, \quad A' = B, \text{ and } \quad A = B'.$$

The set A is called the **extent** while the set B is called the **intent** of the formal concept (A, B). ◊

According to this definition, a formal concept has two parts: its extent and its intent. This follows an old tradition in philosophical concept logic, as expressed in the *Logic of Port Royal, 1654* [2], and in the International Standard ISO 704 (*Terminology work – Principles and methods*, translation of the German Standard DIN 2330).

The description of a concept by extent and intent is redundant, because each of the two parts determines the other (since $B = A'$ and $A = B'$). But for many reasons this redundant description is very convenient.

When a formal context is written as a cross table, then every formal concept (A, B) corresponds to a (filled) rectangular subtable, with row set A and column set B. To make this more precise, note that in the definition of a formal context, there is no order on the sets G or M. Permuting the rows or the columns of a cross table therefore does not change the formal context it represents. A *rectangular subtable* may, in this sense, omit some rows or columns; it must be rectangular after an appropriate rearrangement of the rows and the columns. It is then easy to characterize the rectangular subtables that correspond to formal concepts: they are full of crosses and maximal with respect to this property.

Lemma 2 *(A, B) is a formal concept of (G,M,I) iff $A \subseteq G$, $B \subseteq M$, and A and B are each maximal (with respect to set inclusion) for the property $A \times B \subseteq I$.*

A formal context may have many formal concepts. In fact, it is not difficult to come up with examples where the number of formal concepts is exponential in the size of the formal context. The set of all formal concepts of (G, M, I) is denoted

$$\mathfrak{B}(G, M, I),$$

or just $\mathfrak{B}$ if the context is known and fixed. Later on we shall discuss an algorithm to compute all formal concepts of a given formal context.

3.4 Conceptual Hierarchy

Formal concepts can be (partially) ordered in a natural way. Again, the definition is inspired by the way we usually order concepts in a *subconcept–superconcept hierarchy*: “Dog” is a subconcept of “mammal”, because every dog is a mammal. Transferring this to formal concepts, the natural definition is as follows:

Definition 6 Let (A_1, B_1) and (A_2, B_2) be formal concepts of (G, M, I). We say that (A_1, B_1) is a **subconcept** of (A_2, B_2) (and, equivalently, that (A_2, B_2) is a **superconcept** of (A_1, B_1)) iff $A_1 \subseteq A_2$. We use the $\leq$-sign to express this relation and thus have

$$(A_1, B_1) \leq (A_2, B_2) :\iff A_1 \subseteq A_2.$$

The set $\mathfrak{B}$ of all formal concepts of (G, M, I), ordered by the relation $\leq$ – that is, the structure $(\mathfrak{B}, \leq)$ – is denoted

$$\underline{\mathfrak{B}}(G, M, I)$$

and is called the **concept lattice** of the formal context (G, M, I). ♢

We will see in a bit, why the structure is called *lattice*. Arguably, this definition is natural, but irritatingly asymmetric. What about the intents? Well, a look at Proposition 1 shows that for concepts (A_1, B_1) and (A_2, B_2)

$$A_1 \subseteq A_2 \quad \text{is equivalent to} \quad B_2 \subseteq B_1.$$

Therefore

$$(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2 \quad (\Longleftrightarrow B_2 \subseteq B_1).$$

The concept lattice of a formal context is a partially ordered set. We recall the formal definition of such a partial ordered set in the following.

Definition 7 A **partially ordered set** is a pair $(P, \leq)$ where P is a set, and $\leq$ is a binary relation on P (i. e., $\leq$ is a subset of $P \times P$) which is

1. reflexive ($x \leq x$ for all $x \in P$),
2. anti-symmetric ($x \leq y$ and $y \leq x$ imply $x = y$ for all $x, y \in P$), and
3. transitive ($x \leq y$ and $y \leq z$ imply $x \leq z$ for all $x, y, z \in P$).

We write $x \geq y$ for $y \leq x$, and $x < y$ for $x \leq y$ with $x \neq y$. $\Diamond$

Partially ordered sets appear frequently in mathematics and computer science. Observe that we do not assume a **total order**, which would require the additional condition $x \leq y$ or $y \leq x$ for all $x, y \in P$. Concept lattices have additional properties beyond being partially ordered sets, that is why we call them 'lattices'. This will be the topic of the next section.

3.5 Concept lattice diagrams

The concept lattice of (G, M, I) is the set of all formal concepts of (G, M, I), ordered by the subconcept–superconcept order. Ordered sets of moderate size can conveniently be displayed as **order diagram**s, sometimes also referred to as line diagrams. We explain how to *read* such a concept lattice line diagram by means of an example given in Figure 4. Later on, we will discuss how to *draw* such diagrams.

Figure 4 refers to the following situation: Think of two squares of equal size that are drawn on paper. There are different ways to arrange the two squares: they may be disjoint (i. e., have no point in common), may overlap (i. e., have a common interior point), may share a vertex, an edge or a line segment of the boundary (of length > 0), they may be parallel or not.

Figure 4 shows a concept lattice unfolding these possibilities. It consists of twelve formal concepts, represented by the twelve small circles in the diagram. The names of the six attributes are given. Each name is attached to one of the formal concepts and is written slightly above the respective circle. The ten objects are represented by little pictures; each showing a pair of unit squares. Again, each object is attached to exactly one formal concept; the picture representing the object is drawn slightly below the circle representing the object concept.

Some of the circles are connected by edges. These express the concept order. With the help of the edges, we can read from the diagram which concepts are subconcepts of which other concepts, and which objects have which attributes. To do so, one has to follow *ascending paths* in the diagram.

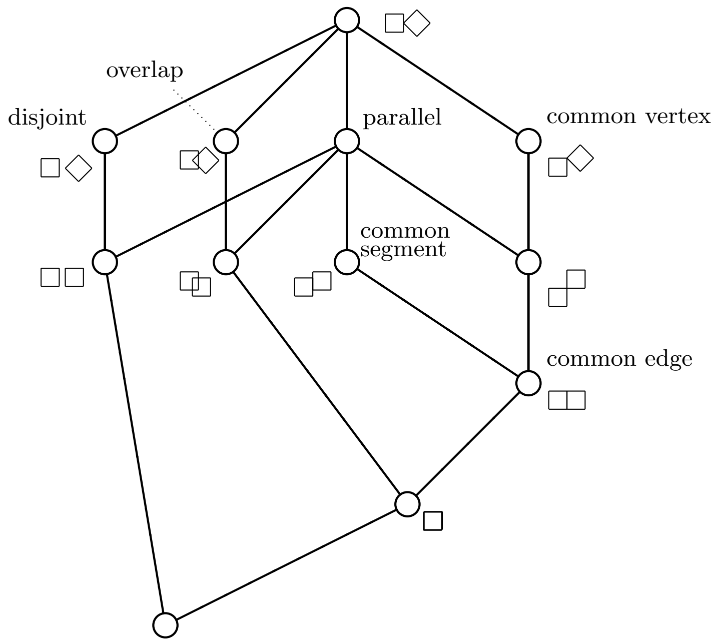


Fig. 4. A concept lattice diagram. The objects are pairs of unit squares. The attributes describe their mutual position.

For example, consider the object □□. From the corresponding circle we can reach, via ascending paths, four attributes: “common edge”, “common segment”, “common vertex”, and “parallel”. □□ does in fact have these properties, and does not have any of the others: the two squares are neither “disjoint” nor do they “overlap”.

Similarly, we can find those objects that have a given attribute by following all descending paths starting at the attribute concept. For example, to find all objects which “overlap”, we start at the attribute concept labeled “overlap” and follow the edges downward. We can reach three objects (namely □◇, ⧉, and □, the latter symbolizing two squares at the same position). Note that we cannot reach □ □, because only at concept nodes it is allowed to make a turn.

With the same method, we can read the intent and the extent of every formal concept in the diagram. For example, consider the concept circle labeled ⧉. Its extent consists of all objects that can be reached from that circle on an descending path. The extent therefore is {⧉, □}. Similarly, we find by an inspection of the ascending paths that the intent of this formal concept is {overlap, parallel}.

The diagram contains all necessary information. We can read off the objects, the attributes, and the incidence relation I. Thus we can perfectly reconstruct the formal context (i. e., "the original data") from the diagram.[6] Moreover, for each formal concept we can easily determine its extent and intent from the diagram.

So in a certain sense, concept lattice diagrams are perfect. But there are, of course, limitations. Take another look at Figure 4. Is it *correct*? Is it *complete*? The answer is that, since a concept lattice faithfully unfolds the formal context, the information displayed in the lattice diagram can be only as correct and complete as the formal context is. In our specific example it is easy to check that the given examples in fact do have the properties as indicated. But a more difficult problem is if our selection of objects is representative. Are there possibilities to combine two squares, that lead to an attribute combination not occurring in our sample? We shall come back to that question later.

3.6 Supremum and Infimum

Can we compute with formal concepts? Yes, we can. The concept operations are however quite different from addition and multiplication of numbers. They resemble more of the operations *greatest common divisor* and *least common multiple*, that we know from integers.

Definition 8 Let $(M, \leq)$ be a partially ordered set, and A be a subset of M. A **lower bound** of A is an element s of M with $s \leq a$, for all $a \in A$. An **upper bound** of A is defined dually. If there exists a largest element in the set of all lower bounds of A, then it is called the **infimum** (or **meet**) of A. It is denoted *inf* A or $\bigwedge A$. The **supremum** (or **join**) of A (*sup* A, $\bigvee A$) is defined dually. For $A = \{x, y\}$, we write also $x \wedge y$ for their infimun, and $x \vee y$ for their supremum. $\Diamond$

Lemma 3 *For any two formal concepts* (A_1, B_1) *and* (A_2, B_2) *of some formal context we obtain*

- *the infimum (***greatest common subconcept***) of* (A_1, B_1) *and* (A_2, B_2) *as*

$$(A_1, B_1) \wedge (A_2, B_2) := (A_1 \cap A_2, (B_1 \cup B_2)''),$$

- *the supremum (***least common superconcept***) of* (A_1, B_1) *and* (A_2, B_2) *as*

$$(A_1, B_1) \vee (A_2, B_2) := ((A_1 \cup A_2)'', B_1 \cap B_2).$$

It is not difficult to prove that what is suggested by this definition is in fact true: $(A_1, B_1) \wedge (A_2, B_2)$ is in fact a formal concept (of the same context), $(A_1, B_1) \wedge (A_2, B_2)$ is a subconcept of both (A_1, B_1) and (A_2, B_2), and any other common subconcept of (A_1, B_1) and (A_2, B_2) is also a subconcept of $(A_1, B_1) \wedge (A_2, B_2)$. Similarly, $(A_1, B_1) \vee (A_2, B_2)$ is a formal concept, it is a superconcept of (A_1, B_1) and of (A_2, B_2), and it is a subconcept of any common superconcept of these two formal concepts.

[6] This reconstruction is assured by the Basic Theorem given below.

With some practice, one can read off infima and suprema from the lattice diagram. Choose any two concepts from Figure 4 and follow the descending paths from the corresponding nodes in the diagram. There is always a highest point where these paths meet, that is, a highest concept that is below both, namely, the infimum. Any other concept below both can be reached from the highest one on a descending path. Similarly, for any two formal concepts there is always a lowest node (the supremum of the two), that can be reached from both concepts via ascending paths. And any common superconcept of the two is on an ascending path from their supremum.

3.7 Complete lattices

The operations for computing with formal concepts, infimum and supremum, are not as weird as one might suspect. In fact, we obtain with each concept lattice an algebraic structure called a "lattice", and such structures occur frequently in mathematics and computer science. "Lattice theory" is an active field of research in mathematics. A *lattice* is an algebraic structure with two operations (called "meet" and "join" or "infimum" and "supremum") that satisfy certain natural conditions:[7]

Definition 9 A partially ordered set $\mathbb{V} := (V, \leq)$ is called a **lattice**, if their exists, for every pair of elements $x, y \in V$, their infimum $x \wedge y$ as well as their supremum $x \vee y$. $\Diamond$

We shall not discuss the algebraic theory of lattices in this lecture. Many universities offer courses in lattice theory, and there are excellent textbooks.[8]

Concept lattices have an additional nice property: they are **complete lattices**. This means that the operations of infimum and supremum do not only work for an input consisting of two elements, but for arbitrary many. In other words: each collection of formal concepts has a greatest common subconcept and a least common superconcept. This is even true for infinite sets of concepts. The operations "infimum" and "supremum" are not necessarily binary, they work for any input size.

Definition 10 A partially ordered set $\mathbb{V} := (V, \leq)$ is a **complete lattice**, if for every set $A \subseteq V$, there exists its infimum $\bigwedge V$ and its supremum $\bigvee A$. $\Diamond$

Note that the definition requests the existence of infimum and supremum for every set A, hence also for the empty set $A := \emptyset$. Following the definition, we obtain that $\bigwedge \emptyset$ has to be the (unique) largest element of the lattice. It is denoted by $\mathbf{1_V}$. Dually, $\bigvee \emptyset$ has to be the smallest element of the lattice; it is denoted by $\mathbf{0_V}$.

[7] Unfortunately, the word "lattice" is used with different meanings in mathematics. It also refers to generalized grids.

[8] *An introduction to lattices and order* by B. Davey and H. Priestley is particularly popular among CS students.

The arbitrary arity of infimum and supremum is very useful, but will make essentially no difference for our considerations, because we shall mainly be concerned with finite formal contexts and finite concept lattices. Well, this is not completely true. In fact, although the concept lattice in Figure 4 is finite, its ten objects are representatives for *all* possibilities to combine two unit squares. Of course, there are infinitely many such possibilities. It is true that we shall consider finite concept lattices, but our examples may be taken from an infinite reservoir.

3.8 The Basic Theorem of FCA

We give now a mathematically precise formulation of the algebraic properties of concept lattices. The theorem below is not difficult, but basic for many other results. Its formulation contains some technical terms that we have not mentioned so far.

In a complete lattice, an element is called **supremum-irreducible** if it cannot be written as a supremum of other elements, and **infimum-irreducible** if it can not be expressed as an infimum of other elements. It is very easy to locate the irreducible elements in a diagram of a finite lattice: the supremum-irreducible elements are precisely those from which there is exactly one edge going downward. An element is infimum-irreducible if and only if it is the start of exactly one upward edge. In Figure 4, there are precisely nine supremum-irreducible concepts and precisely five infimum-irreducible concepts. Exactly four concepts have both properties, they are **doubly irreducible**.

A set of elements of a complete lattice is called **supremum-dense**, if every lattice element is a supremum of elements from this set. Dually, a set is called **infimum-dense**, if the infima that can be computed from this set exhaust all lattice elements.

The notion of isomorphism defined next essentially captures the idea of two lattices being the same up to a renaming of the elements.

Definition 11 Two lattices $\mathbb{V}$ and $\mathbb{W}$ are **isomorphic** ($\mathbb{V} \cong \mathbb{W}$), if there exists a bijective mapping $\varphi\colon V \to W$ with $x \leq y \iff \varphi(x) \leq \varphi(y)$. The mapping φ is then called **lattice isomorphism** between $\mathbb{V}$ and $\mathbb{W}$. $\Diamond$

Now we have defined all the terminology necessary for stating the main theorem of Formal Concept Analysis.

Theorem 4 (The Basic Theorem of Formal Concept Analysis.) *The concept lattice of any formal context (G, M, I) is a complete lattice. For an arbitrary set $\{(A_i, B_i) \mid i \in J\} \subseteq \mathfrak{B}(G, M, I)$ of formal concepts, the supremum is given by*

$$\bigvee_{i \in J} (A_i, B_i) = \left((\bigcup_{i \in J} A_i)'', \bigcap_{i \in J} B_i \right)$$

and the infimum is given by

$$\bigwedge_{i\in J}(A_i, B_i) = \left(\bigcap_{i\in J} A_i, (\bigcup_{i\in J} B_i)''\right).$$

A complete lattice L is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$ precisely if there are mappings $\tilde{\gamma} : G \to L$ and $\tilde{\mu} : M \to L$ such that $\tilde{\gamma}(G)$ is supremum-dense and $\tilde{\mu}(M)$ is infimum-dense in L, and for all $g \in G$ and $m \in M$

$$g\ I\ m \iff \tilde{\gamma}(g) \leq \tilde{\mu}(m).$$

In particular, $L \cong \underline{\mathfrak{B}}(L, L, \leq)$.

The theorem is less complicated than it may first seem. We give some explanations below. Readers in a hurry may skip these and continue with the next section.

The first part of the theorem gives the precise formulation for infimum and supremum of arbitrary sets of formal concepts. The second part of the theorem gives (among other information) an answer to the question if concept lattices have any special properties. The answer is "no": every complete lattice is (isomorphic to) a concept lattice. This means that for every complete lattice, we must be able to find a set G of objects, a set M of attributes and a suitable relation I, such that the given lattice is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$. The theorem does not only say how this can be done, it describes in fact *all* possibilities to achieve this.

In Figure 4, every object is attached to a unique concept, the corresponding *object concept*. Similarly for each attribute there corresponds an *attribute concept*. These can be defined as follows:

Definition 12 Let (G, M, I) be some formal context. Then

– for each object $g \in G$ the corresponding **object concept** is

$$\gamma g := (\{g\}'', \{g\}'),$$

– and for each attribute $m \in M$ the **attribute concept** is given by

$$\mu m := (\{m\}', \{m\}'').$$

The set of all object concepts of (G, M, I) is denoted γG, the set of all attribute concepts is μM.
$\Diamond$

Using Definition 5 and Proposition 1, it is easy to check that these expressions in fact define formal concepts of (G, M, I).

We have that $\gamma g \leq (A, B) \iff g \in A$. A look at the first part of the Basic Theorem shows that each formal concept is the supremum of all the object concepts below it. Therefore, the set γG of all object concepts is supremum-dense. Dually, the attribute concepts form an infimum-dense set in $\underline{\mathfrak{B}}(G, M, I)$.

The Basic Theorem says that, conversely, any supremum-dense set in a complete lattice L can be taken as the set of objects and any infimum-dense set be taken as a set of attributes for a formal context with concept lattice isomorphic to L.

We conclude with a simple observation that often helps to find errors in concept lattice diagrams. The fact that the object concepts form a supremum-dense set implies that every supremum-irreducible concept must be an object concept (the converse is not true). Dually, every infimum-irreducible concept must be an attribute concept. This yields the following rule for concept lattice diagrams:

Proposition 5 *Given a formal context (G, M, I) and a finite order diagram, labeled by the objects from G and the attributes from M. For $g \in G$ let $\tilde{\gamma}(g)$ denote the element of the diagram that is labeled with g, and let $\tilde{\mu}(m)$ denote the element labeled with m. Then the given diagram is a correctly labeled diagram of $\underline{\mathfrak{B}}(G, M, I)$ if and only if it satisfies the following conditions:*

1. *The diagram is a correct lattice diagram,*
2. *every supremum-irreducible element is labeled by some object,*
3. *every infimum-irreducible element is labeled by some attribute,*
4. *$g\ I\ m \iff \tilde{\gamma}(g) \leq \tilde{\mu}(m)$ for all $g \in G$ and $m \in M$.*

The definitions of lattices and complete lattices are self-dual: If $(V, \leq)$ is a (complete) lattice, then $(V, \leq)^d := (V, \geq)$ is also a (complete) lattice. If a theorem holds for a (complete) lattice, then the 'dual theorem' also holds, i. e., the theorem where all occurences of $\leq, \vee, \wedge, \bigvee, \bigwedge, \mathbf{0_V}, \mathbf{1_V}$ etc. are replaced by $\geq, \wedge, \vee, \bigwedge, \bigvee, \mathbf{1_V}, \mathbf{0_V}$, resp.

For concept lattices, their dual can be obtained by "flipping" the formal context:

Lemma 6 *Let (G, M, I) be a formal context and $\underline{\mathfrak{B}}(G, M, I)$ its concept lattice. Then $(\underline{\mathfrak{B}}(G, M, I))^d \cong \underline{\mathfrak{B}}(M, G, I^{-1})$, with $I^{-1} := \{(m, g) \mid (g, m) \in I\}$.*

3.9 Computing all Concepts of a Context

There are several algorithms that help drawing concept lattices. We shall discuss some of them below. But we find it instructive to start by some small examples that can be drawn by hand. For computing concept lattices, we will investigate a fast algorithm later. We start with a naive method before proceeding to a method which is suitable for manual computation.

In principle, it is not difficult to find all the concepts of a formal context. The following proposition summarizes the naive possibilities of generating all concepts.

Lemma 7 *Each concept of a context (G, M, I) has the form (X'', X') for some subset $X \subseteq G$ and the form (Y', Y'') for some subset $Y \subseteq M$. Conversely, all such pairs are concepts. Every extent is the intersection of attribute extents and every intent is the intersection of object intents.*

The first part of the lemma suggests a first algorithm for computing all concepts: go through all subsets X of G and record (X'', X') as concept (skipping duplicates). However, this is rather inefficient, and not practicable even for relatively small contexts. The second part of the proposition at least yields the possibility to calculate the concepts of a small context by hand.

The following method is more efficient, and is recommended for computations by hand. It is based on the following observations:

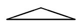
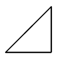
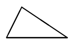
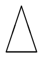
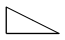
1. It suffices to determine all concept extents (or all concept intents) of (G, M, I), since we can always determine the other part of a formal concept with the help of the derivation operators.
2. The intersection of arbitrary many extents is an extent (and the intersection of arbitrary intents is an intent). This follows easily from the formulae given in the Basic Theorem. By the way: a convention that may seem absurd on the first glance allows to include in "arbitrary many" also the case "zero". The convention says that the intersection of zero intents equals M and the intersection of zero extents equals G.
3. One can determine all extents from knowing all **attribute extent**s $\{m\}'$, $m \in M$ (and all intents from all **object intent**s $\{g\}'$, $g \in G$) because every extent is an intersection of attribute extents (and every intent is the intersection of object intents). This follows from the fact that the attribute concepts are infimum-dense and the object concepts are supremum-dense.

These observations give rise to the following procedure.

Instruction for determining all formal concepts of a small formal context

1. Initialize a list of concept extents. To begin with, write for each attribute $m \in M$ the attribute extent $\{m\}'$ to this list (if not already present).
2. For any two sets in this list, compute their intersection. If the result is a set that is not yet in the list, then extend the list by this set. With the extended list, continue to build all pairwise intersections.
3. If for any two sets in the list their intersection is also in the list, then extend the list by the set G (provided it is not yet contained in the list). The list then contains all concept extents (and nothing else).
4. For every concept extent A in the list compute the corresponding intent A' to obtain a list of all formal concepts (A, A') of (G, M, I).

Example 1 We illustrate the method by means of an example from elementary geometry. The objects of our example are seven triangles. The attributes are five standard properties that triangles may or may not have:

Triangles		
abbreviation	coordinates	diagram
T1	(0,0) (6,0) (3,1)	
T2	(0,0) (1,0) (1,1)	
T3	(0,0) (4,0) (1,2)	
T4	(0,0) (2,0) $(1,\sqrt{3})$	
T5	(0,0) (2,0) (5,1)	
T6	(0,0) (2,0) (1,3)	
T7	(0,0) (2,0) (0,1)	

Attributes	
symbol	property
a	equilateral
b	isoceles
c	acute angled
d	obtuse angled
e	right angled

We obtain the following formal context

	a	b	c	d	e
$T1$		$\times$		$\times$	
$T2$		$\times$			$\times$
$T3$			$\times$		
$T4$	$\times$	$\times$	$\times$		
$T5$				$\times$	
$T6$		$\times$	$\times$		
$T7$					$\times$

Following the above instruction, we proceed:

1. Write the attribute extents to a list.

No. extent	found as
$e_1 := \{T_4\}$	$\{a\}'$
$e_2 := \{T_1, T_2, T_4, T_6\}$	$\{b\}'$
$e_3 := \{T_3, T_4, T_6\}$	$\{c\}'$
$e_4 := \{T_1, T_5\}$	$\{d\}'$
$e_5 := \{T_2, T_7\}$	$\{e\}'$

2. Compute all pairwise intersections, and

3. add G.

No. extent	found as
$e_1 := \{T_4\}$	$\{a\}'$
$e_2 := \{T_1, T_2, T_4, T_6\}$	$\{b\}'$
$e_3 := \{T_3, T_4, T_6\}$	$\{c\}'$
$e_4 := \{T_1, T_5\}$	$\{d\}'$
$e_5 := \{T_2, T_7\}$	$\{e\}'$
$e_6 := \emptyset$	$e_1 \cap e_4$
$e_7 := \{T_4, T_6\}$	$e_2 \cap e_3$
$e_8 := \{T_1\}$	$e_2 \cap e_4$
$e_9 := \{T_2\}$	$e_2 \cap e_5$
$e_{10} := \{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$	step 3

4. Compute the intents.

Concept No.	(extent , intent)
1	$(\{T_4\}\ ,\ \{a, b, c\})$
2	$(\{T_1, T_2, T_4, T_6\}\ ,\ \{b\})$
3	$(\{T_3, T_4, T_6\}\ ,\ \{c\})$
4	$(\{T_1, T_5\}\ ,\ \{d\})$
5	$(\{T_2, T_7\}\ ,\ \{e\})$
6	$(\emptyset\ ,\ \{a, b, c, d, e\})$
7	$(\{T_4, T_6\}\ ,\ \{b, c\})$
8	$(\{T_1\}\ ,\ \{b, d\})$
9	$(\{T_2\}\ ,\ \{b, e\})$
10	$(\{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}\ ,\ \emptyset)$

We have now computed all ten formal concepts of the triangles–context. The last step can be skipped if we are not interested in an explicit list of all concepts, but just in computing a line diagram.

3.10 Drawing Concept Lattices

Based on one of the lists 3. or 4., we can start to draw a diagram. Before doing so, we give two simple definitions.

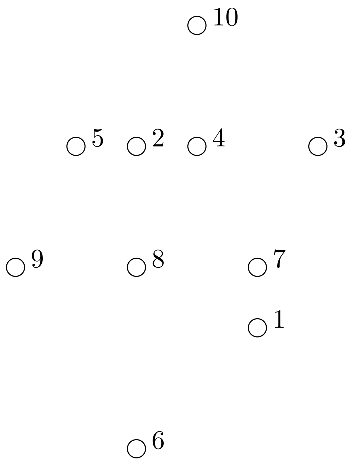
Definition 13 Let (A_1, B_1) and (A_2, B_2) be formal concepts of some formal context (G, M, I). We say that (A_1, B_1) is a **proper subconcept** of (A_2, B_2) (written as $(A_1, B_1) < (A_2, B_2)$), if $(A_1, B_1) \leq (A_2, B_2)$ and $(A_1, B_1) \neq (A_2, B_2)$. We call (A_1, B_1) a **lower neighbour** of (A_2, B_2) (written as $(A_1, B_1) \prec (A_2, B_2)$), if $(A_1, B_1) < (A_2, B_2)$, but no formal concept (A, B) of (G, M, I) exists with $(A_1, B_1) < (A, B) < (A_2, B_2)$. $\diamondsuit$

Instruction how to draw a line diagram of a small concept lattice

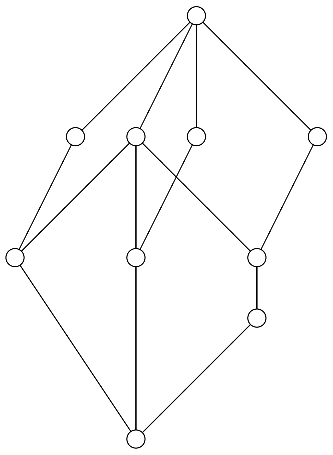
5. Take a sheet of paper and draw a small circle for every formal concept, in the following manner: a circle for a concept is always positioned higher than the all circles for its proper subconcepts.
6. Connect each circle with the circles of its lower neighbors.
7. Label with attribute names: attach the attribute m to the circle representing the concept $(\{m\}', \{m\}'')$.
8. Label with object names: attach each object g to the circle representing the concept $(\{g\}'', \{g\}')$.

We now follow these instructions.

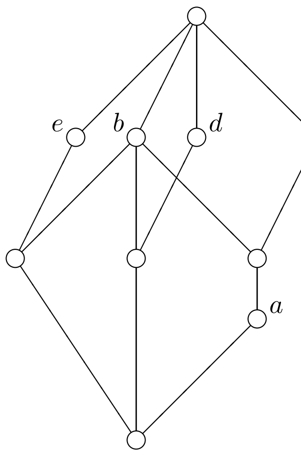
5. Draw a circle for each of the formal concepts:



6. Connect circles with their lower neighbours:



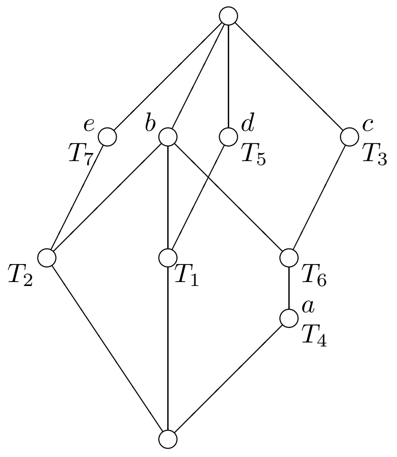
7. Add the attribute names:



8. Determine the object concepts

object g	object intent $\{g\}'$	no. of concept
T_1	$\{b, d\}$	8
T_2	$\{b, e\}$	9
T_3	$\{c\}$	3
T_4	$\{a, b, c\}$	1
T_5	$\{d\}$	4
T_6	$\{b, c\}$	7
T_7	$\{e\}$	5

and add the object names to the diagram:



Done! Usually it takes some attempts before a nice, readable diagram is achieved. Finally we can make the effort to avoid abbreviations and to increase the readability. The result is shown in Figure 5.

3.11 Clarifying and Reducing a Formal Context

There are context manipulations that simplify a formal context without changing the diagram, except for the labeling. It is usually advisable to do these manipulations first, before starting computations.

The simplest operation is **clarification**, which refers to identifying "equal rows" of a formal context, and "equal columns" as well. What is meant is that if a context contains objects $g_1, g_2, \ldots$ with $\{g_i\}' = \{g_j\}'$ for all i, j, that is, objects which have exactly the same attributes, then these can be replaced by a single object, the name of which is just the list of names of these objects. The same can be done for attributes with identical attribute extent.

Definition 14 We say that a formal context is **clarified** if no two of its object intents are equal and no two of its attribute extents are equal. $\Diamond$

A stronger operation is **reduction**, which refers to omitting attributes that are equivalent to combinations of other attributes (and dually for objects). For defining reduction it is convenient to work with a clarified context.

Definition 15 An attribute m of a clarified context is called **reducible** if there is a set $S \subseteq M$ of attributes with $\{m\}' = S'$, otherwise it is **irreducible**. Reduced objects are defined dually. A formal context is called **reduced**, if all objects and all attribues are irreducible. $\Diamond$

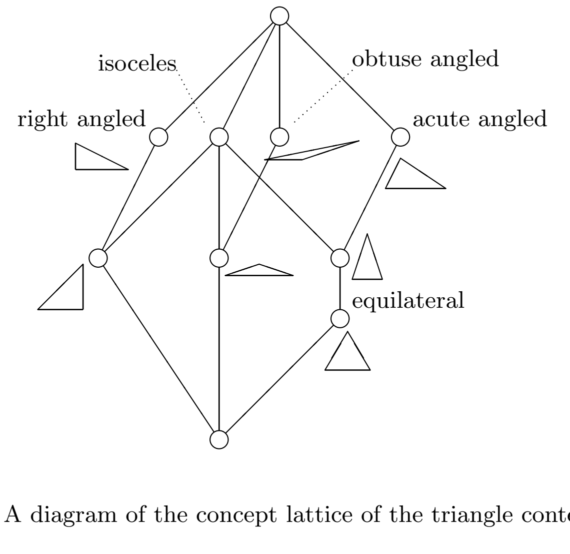


Fig. 5. A diagram of the concept lattice of the triangle context.

$\{m\}' = S'$ means that an object g has the attribute m if and only if it has all the attributes from S. If we delete the column m from our cross table, no essential information is lost because we can reconstruct this column from the data contained in other columns (those of S). Moreover, deleting that column does not change the number of concepts, nor the concept hierarchy, because $\{m\}' = S'$ implies that m is in the intent of a concept if and only if S is contained in that intent. The same is true for reducible objects and concept extents. Deleting a reducible object from a formal context does not change the structure of the concept lattice.

It is even possible to remove several reducible objects and attributes simultaneously from a formal context without any effect on the lattice structure, as long as the number of removed elements is finite.

Definition 16 Let (G, M, I) be a finite context, and let G_{irr} be the set of irreducible objects and M_{irr} be the set of irreducible attributes of (G, M, I). The context $(G_{irr}, M_{irr}, I \cap G_{irr} \times M_{irr})$ is the **reduced context** corresponding to (G, M, I).

For a finite lattice L let $J(L)$ denote the set of its supremum-irreducible elements and let $M(L)$ denote the set of its infimum-irreducible elements. Then $(J(L), M(L), \leq)$ is the **standard context** for the lattice L. $\Diamond$

Proposition 8 *A finite context and its reduced context have isomorphic concept lattices. For every finite lattice L there is (up to isomorphism) exactly one reduced context, the concept lattice of which is isomorphic to L, namely its standard context.*

3.12 Additive and Nested Line Diagrams

In this section, we discuss possibilities to generate line diagrams both automatically or by hand. A list of some dozens of concepts may already be quite difficult to survey, and it requires practice to draw good line diagrams of concept lattices with more than 20 elements.

The best and most versatile form of representation for a concept lattice is a well-drawn line diagram. It is, however, tedious to draw such a diagram by hand and one would wish an automatic generation by means of a computer. We know quite a few algorithms to do this, but none which provides a general satisfactory solution. It is by no means clear which qualities make up a *good* diagram. It should be transparent, easily readable and should facilitate the interpretation of the data represented. How this can be achieved in each individual case depends, however, on the aim of the interpretation and on the structure of the lattice. Simple optimization criteria (minimization of the number of edge crossings, drawing in layers, etc.) often bring about results that are unsatisfactory. Nevertheless, automatically generated diagrams are a great help: they can serve as the starting point for drawing by hand. Therefore, we will describe simple methods of generating and manipulating line diagrams by means of a computer.

3.12.1 Additive Line Diagrams We will now explain a method where a computer generates a diagram and offers the possibility of improving it interactively. Programming details are irrelevant in this context. We will therefore only give a **positioning rule** which assigns points in the plane to the elements of a given ordered set $(P, \leq)$. If a and b are elements of P with $a < b$, the point assigned to a must be lower than the point assigned to b (i.e., it must have a smaller y-coordinate). This is guaranteed by our method. We will leave the computation of the edges and the checking for undesired coincidences of vertices and edges to the program. We do not even guarantee that our positioning is injective (which of course is necessary for a correct line diagram). This must also be checked if necessary.

Definition 17 A **set representation** of an ordered set $(P, \leq)$ is an order embedding of $(P, \leq)$ in the power-set of a set X, i.e., a map

$$\operatorname{rep} : P \to \mathfrak{P}(X)$$

with the property

$$x \leq y \iff \operatorname{rep} x \subseteq \operatorname{rep} y.$$

$\Diamond$

An example of a set representation for an arbitrary ordered set $(P, \leq)$ is the assignment

$$X := P, \quad a \mapsto \{x \mid x < a\}.$$

In the case of a concept lattice,

$$X := G, \quad (A, B) \mapsto A$$

is a set representation.

$$X := M, \quad (A, B) \mapsto M \setminus B$$

is another set representation, and both can be combined to

$$X := G \mathbin{\dot\cup} M, \quad (A, B) \mapsto A \cup (M \setminus B).$$

It is sufficient to limit oneself to the irreducible objects and attributes.

For an **additive line diagram** of an ordered set $(P, \leq)$ we need a set representation $\operatorname{rep} : P \to \mathfrak{P}(X)$ as well as a **grid projection**

$$\operatorname{vec} : X \to \mathbb{R}^2,$$

assigning a real vector with a positive y-coordinate to each element of X. By

$$\operatorname{pos} p := n + \sum_{x \in \operatorname{rep} p} \operatorname{vec} x$$

we obtain a positioning of the elements of P in the plane. Here, n is a vector which can be chosen arbitrarily in order to shift the entire diagram. By only allowing positive y–coordinates for the grid projection we make sure that no element p is positioned below an element q with $q < p$.

Every finite line diagram can be interpreted as an additive diagram with respect to an appropriate set representation. For concept lattices we usually use the representation by means of the irreducible objects and/or attributes. The resulting diagrams are characterized by a great number of parallel edges, which improves their readability. Experience shows that the set representation by means of the irreducible attributes is most likely to result in an easily interpretable diagram. Figure 5 for instance was obtaining by selecting the irreducible attributes for the set representation.

Since the second set representation given above is somehow unnatural, we introduce for this purpose the dual set representation.

Definition 18 A **dual set representation** of an ordered set $(P, \leq)$ is an order–inversing embedding of $(P, \leq)$ in the power-set of a set X, i.e., a map

$$\mathrm{rep}': P \to \mathfrak{P}(X)$$

with the property

$$x \leq y \iff \mathrm{rep}'x \supseteq \mathrm{rep}'y.$$

◊

Now

$$X := M, \quad \mathrm{rep}': (A, B) \mapsto B$$

is a dual set representation. We request now that the grid projection allows only negative y–coordinates. The following shows that the two ways are indeed equivalent: Let $\mathrm{vec}': X \to \mathbb{R}^2$ be given by $\mathrm{vec}'(m) := (-x, -y)$ where $\mathrm{vec}(m) = (x, y)$ for all $m \in X$. Then all y–coordinates are indeed negative. We obtain then the following equality:

$$\begin{aligned}\mathrm{pos}(A, B) &= n + \sum_{m \in M \setminus B} \mathrm{vec}(m)\\ &= n + \sum_{m \in M} \mathrm{vec}(m) + \sum_{m \in B} - \mathrm{vec}(m)\\ &= n' + \sum_{m \in B} \mathrm{vec}'(m)\\ &= \mathrm{pos}'(A, B)\end{aligned}$$

where $n' := n + \sum_{m \in M} \mathrm{vec}(m)$.

It is particularly easy to manipulate these diagrams: If we change – the set representation being fixed – the grid projection for an element $x \in X$, this means that all images of the order filter $\{p \in P \mid x \in \mathrm{rep}\, p\}$ are shifted by the same distance and that all other points remain in the same position. In the case of the set representation by means of the irreducibles these order filters are precisely principal filters or complements of principal ideals, respectively. This means that we can manipulate the diagram by shifting principal filters or principal ideals, respectively, and leaving all other elements in position.

Even carefully constructed line diagrams loose their readability from a certain size up, as a rule from around 50 elements up. One gets considerably further with *nested line diagrams* which will be introduced next. However, these diagrams do not only serve to represent larger concept lattices. They offer the possibility to visualize how the concept lattice changes if we add further attributes.

3.12.2 Nested Line Diagrams Nested line diagrams permit a satisfactory graphical representation of somewhat larger concept lattices. The basic idea of the nested line diagram consists of clustering parts of an ordinary diagram and replacing bundles of parallel lines between these parts by one line each. Thus, a nested line diagram consists of ovals, which contain clusters of the ordinary line diagram and which are connected by lines. In the simplest case, two ovals which are connected by a simple line are congruent. Here, the line indicates that corresponding circles within the ovals are direct neighbors, resp.

Furthermore, we allow that two ovals connected by a single line do not necessarily have to be congruent, but they may each contain a part of two congruent figures. In this case, the two congruent figures are drawn in the ovals as a "background structure", and the elements are drawn as solid circles if they are part of the respective substructures. The line connecting the two boxes then indicates that the respective pairs of elements of the background shall be connected with each other. An example is given in Figure 6. It is a screenshot of a library information system which was set up for the library of the Center on Interdisciplinary Technology Research of Darmstadt University of Technology.

Nested line diagrams originate from partitions of the set of attributes. The basis is the following theorem:

Theorem 9 *Let (G, M, I) be a context and $M = M_1 \cup M_2$. The map*

$$(A, B) \mapsto (((B \cap M_1)', B \cap M_1), ((B \cap M_2)', B \cap M_2))$$

is a supremum-preserving order embedding of $\mathfrak{B}(G, M, I)$ in the direct product of $\underline{\mathfrak{B}}(G, M_1, I \cap G \times M_1)$ and $\underline{\mathfrak{B}}(G, M_2, I \cap G \times M_2)$. The component maps

$$(A, B) \mapsto ((B \cap M_i)', B \cap M_i)$$

are surjective on $\underline{\mathfrak{B}}(G, M_i, I \cap G \times M_i)$.

In order to sketch a nested line diagram, we proceed as follows: First of all, we split up the attribute set: $M = M_1 \cup M_2$. This splitting up does not have to be disjoint. More important for interpretation purposes is the idea that the sets M_i bear meaning. Now, we draw line diagrams of the subcontexts $\mathbb{K}_i := (G, M_i, I \cap G \times M_i)$, $i \in \{1, 2\}$ and label them with the names of the objects and attributes, as usual. Then we sketch a nested diagram of the product of the concept lattices $\underline{\mathfrak{B}}(\mathbb{K}_i)$ as an auxiliary structure. For this purpose, we draw

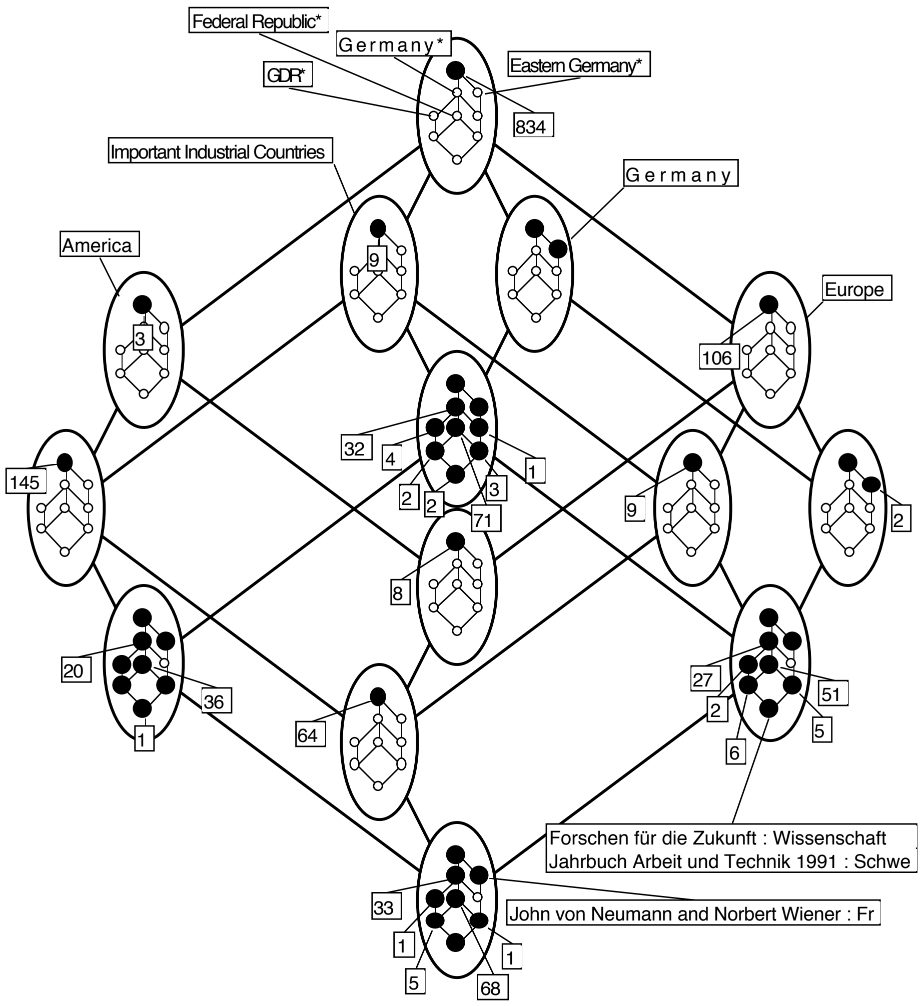

Federal Republic*
Germany*
Eastern Germany*
GDR*
834
Important Industrial Countries
Germany
America
9
3
Europe
106
145
32
4
1
2
2
3
71
9
2
8
20
36
1
27
51
2
5
6
64
Forschen für die Zukunft : Wissenschaft
Jahrbuch Arbeit und Technik 1991 : Schwe
33
John von Neumann and Norbert Wiener : Fr
1
1
5
68

Fig. 6. Nested line diagram of a library information system

a large copy of the diagram of $\underline{\mathfrak{B}}(\mathbb{K}_1)$, representing the lattice elements not by small circles but by congruent ovals, which contain each a diagram of $\underline{\mathfrak{B}}(\mathbb{K}_2)$.

By Theorem 9 the concept lattice $\underline{\mathfrak{B}}(G, M, I)$ is embedded in this product as a $\bigvee$-semilattice. If a list of the elements of $\underline{\mathfrak{B}}(G, M, I)$ is available, we can enter them into the product according to their intents. If not, we enter the object concepts the intents of which can be read off directly from the context, and form all suprema.

This at the same time provides us with a further, quite practicable method of determining a concept lattice by hand: split up the attribute set as appropriate, determine the (small) concept lattices of the subcontexts, draw their product in form of a nested line diagram, enter the object concepts and close it against suprema. This method is particularly advisable in order to arrive at a useful diagram quickly.

4 Closure Systems

The algorithm that will be one central theme of our course was developed for concept lattices, but can be rephrased without reference to Formal Concept Analysis. The reason is that the algorithm essentially relies on a single property of concept lattices, namely that the set of concept intents is closed under intersections. The technique can be formulated for arbitrary intersection closed families of sets, that is, for *closure systems*. Readers who are familiar with closure systems but not with Formal Concept Analysis may prefer this approach.

But note that this means no generalization. We will show that closure systems are not more general than systems of concept intents.

4.1 Definition and examples

Closure systems occur frequently in mathematics and computer science. Their definition is very simple, but not very intuitive when encountered for the first time. The reason is their higher level of abstraction: closure systems are *sets of sets* with certain properties.

Let us recall some elementary notions how to work with sets of sets. For clarity, we shall normally use small latin letters for elements, capital latin letters for sets and calligraphic letters for sets of sets. Given a (nonempty) set $\mathcal{S}$ of sets, we may ask

- which elements occur in these sets? The answer is given by the **union** of $\mathcal{S}$, denoted by
$$\bigcup \mathcal{S} := \{x \mid x \in S \text{ for some } S \in \mathcal{S}\}.$$
- which elements occur in each of these sets? The answer is given by the **intersection** of $\mathcal{S}$, denoted by
$$\bigcap \mathcal{S} := \{x \mid x \in S \text{ for every } S \in \mathcal{S}\}.$$

Some confusion with this definition is caused by the fact that a set of sets may (of course) be empty. Applying the above definition to the case $\mathcal{S} := \emptyset$ is no problem for the union, since

$$\bigcup \emptyset = \{x \mid x \in S \text{ for some } S \in \mathcal{S}\} = \{x \mid \text{false}\} = \emptyset.$$

But there is a problem for the intersection, because the condition "$x \in S$ for every $S \in \mathcal{S}$" is satisfied by *all* x (because there is nothing to be satisfied). But there is no *set of all x*; such sets are forbidden in set theory, because they would lead to contradictions.

For the case $\mathcal{S} = \emptyset$ the intersection is defined only with respect to some base set M. If we work with the subsets of some specified set M (as we often do, for example with the set of all attributes of some formal context), then we define

$$\bigcap \emptyset := M.$$

A set M with, say, n elements, has 2^n subsets. The set of all subsets of a set M is denoted $\mathfrak{P}(M)$ and is called the **power set** of the set M. To indicate that $\mathcal{S}$ is a set of subsets of M, we may therefore simply write $\mathcal{S} \subseteq \mathfrak{P}(M)$.

A closure system on a set M is a set of subsets that contains M and is closed under intersections.

Definition 19 A **closure system** on a set M is a set $\mathcal{C} \subseteq \mathfrak{P}(M)$ satisfying

- $M \in \mathcal{C}$, and
- if $\mathcal{D} \subseteq \mathcal{C}$, then $\bigcap \mathcal{D} \in \mathcal{C}$. ◊

Definition 20 A **closure operator** φ on M is a map $\mathfrak{P}(M) \to \mathfrak{P}(M)$ assigning a **closure** $\varphi X \subseteq M$ to each set $X \subseteq M$, which is

monotone: $X \subseteq Y \Rightarrow \varphi X \subseteq \varphi Y$,
extensive: $X \subseteq \varphi X$, and
idempotent: $\varphi\varphi X = \varphi X$.

(Conditions to be satisfied for all $X, Y \subseteq M$.) ◊

Closure operators are frequently met: their axioms describe the natural properties of a *generating process*. We start with some generating set X, apply the generating process and obtain the generated set, φX, the closure of X. Such generating processes occur in fact in many different variants in mathematics and computer science.

Closure systems and closure operators are closely related. In fact, there is a natural way to obtain from each closure operator a closure system and vice versa. It works as follows:

Lemma 10 *For any closure operator, the set of all closures is a closure system. Conversely, given any closure system $\mathcal{C}$ on M, there is for each subset X of M a unique smallest set $C \in \mathcal{C}$ containing X. Taking this as the closure of X defines a closure operator. The two transformations are inverse to each other.*

Thus closure systems and closure operators are essentially the same. We can add to this:

Theorem 11 *A closure system $\mathcal{C}$ on a set M can be considered as a complete lattice, ordered by set inclusion $\subseteq$. The infimum of any subfamily $\mathcal{D} \subseteq \mathcal{C}$ is equal to $\bigcap \mathcal{D}$, and the supremum is the closure of $\bigcup \mathcal{D}$. Conversely, we can find for any complete lattice L a closure system that is isomorphic to L.*

So closure systems and complete lattices are also very closely related. It comes as no surprise that concept lattices fit well into this relationship. It follows from the Basic Theorem (Thm. 4) that the set of all concept intents of a formal context is closed under intersections and thus is a closure system on M. Dually, the set of all concept extents always is a closure system on G. The corresponding closure operators are just the two operators $X \mapsto X''$ on M and G, respectively.

Conversely, given any closure system $\mathcal{C}$ on a set M, we can construct a formal context such that $\mathcal{C}$ is the set of concept intents. It can be concluded from the Basic Theorem that for example $(\mathcal{C}, M, \ni)$ is such a context. In particular, whenever a closure operator on some set M is considered, we may assume that it is the closure operator $A \mapsto A''$ on the attribute set of some formal context (G, M, I).

Thus, closure systems and closure operators, complete lattices, systems of concept intents, and systems of concept extents: all these are very closely related. It is not appropriate to say that they are "essentially the same", but it is true that all these structures have the same degree of expressiveness; none of them is a generalization of another. A substantial result proved for one of these structures can usually be transferred to the others, without much effort.

4.2 The Next Closure Algorithm

We present a simple algorithm that solves the following task: For a given closure operator on a finite set M, it computes all closed sets.

There are many ways to achieve this. Our algorithm is particularly simple. We shall discuss efficiency considerations below.

We start by endowing our base set M with an arbitrary linear order, so that

$$M = \{m_1 < m_2 < \cdots < m_n\},$$

where n is the number of elements of M. Then every subset $S \subseteq M$ can conveniently be described by its **characteristic vector**

$$\varepsilon_S : M \to \{0, 1\},$$

given by

$$\varepsilon_S(m) := \begin{cases} 1 & \text{if } m \in S \\ 0 & \text{if } m \notin S \end{cases}.$$

For example, if the base set is

$$M := \{a < b < c < d < e < f < g\},$$

then the characteristic vector of the subset $S := \{a, c, d, f\}$ is 1011010. In concrete examples we prefer to write a cross instead of a 1 and a blank or a dot instead of a 0, similarly as in the cross tables representing formal contexts. The characteristic vector of the subset $S := \{a, c, d, f\}$ will therefore be written as

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \times & . & \times & \times & . & \times & . \\ \hline\end{array}.$$

Using this notation, it is easy to see if a given set is a subset of another given set, etc.

The set $\mathfrak{P}(M)$ of all subsets of the base set M is naturally ordered by the subset-order $\subseteq$. This is a complete lattice order, and $(\mathfrak{P}(M), \subseteq)$ is called the **power set lattice** of M. The subset-order is a *partial order*. We can also introduce a *linear* or *total* order of the subsets, for example the **lexicographic** or **lectic order** $\leq$, defined as follows: Let $A, B \subseteq M$ be two distinct subsets. We say that A is *lectically smaller* than B, if the smallest element in which A and B differ belongs to B. Formally,

$$A < B \quad :\Longleftrightarrow \quad \exists i.(i \in B \;\wedge\; i \notin A \;\wedge\; \forall j < i(j \in A \iff j \in B)).$$

For example $\{a, c, e, f\} < \{a, c, d, f\}$, because the smallest element in which the two sets differ is d, and this element belongs to the larger set. This becomes even more apparent when we write the sets as vectors and interprete them as binary numbers:

$$\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ & & & \updownarrow & & & \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \; . \end{array}$$

Note that the lectic order extends the subset-order, i.e.,

$$A \subseteq B \Rightarrow A \leq B.$$

The following notation is helpful:

$$A <_i B \quad :\Longleftrightarrow \quad i \in B \;\wedge\; i \notin A \;\wedge\; \forall j < i\,(j \in A \iff j \in B)).$$

In words: $A <_i B$ iff i is the smallest element in which A and B differ, and $i \in B$.

Proposition 12 *1. $A < B$ if and only if $A <_i B$ for some $i \in M$.*
2. If $A <_i B$ and $A <_j C$ with $i < j$, then $C <_i B$.

We consider a closure operator

$$A \mapsto A''$$

on the base set M. To each subset $A \subseteq M$ it yields[9] its closure $A'' \subseteq M$. Our task is to find a list of all these closures. In principle, we might just follow the definition, compute for each subset $A \subseteq M$ its closure A'' and include that in the list. The problem is that different subsets may have identical closures. So if we want a list that contains each closure *exactly once*, we will have to check many times if a computed closure already exists in the list. Moreover, the number of subsets is exponential: a set with n elements has 2^n subsets. The naive algorithm "for each $A \subseteq M$, compute A'' and check if the result is already listed" therefore requires an exponential number of lookups in a list that may have exponential size.

[9] For our algorithm it is not important *how* the closure is computed.

A better idea is to generate the closures in some predefined order, thereby guaranteeing that every closure is generated only once. The reader may guess that we shall generate the closures in lectic order. We will show how to compute, given a closed set, the *lectically next* one. Then no lookups are necessary. Actually, it will not even be necessary to store the list. For many applications it will suffice to generate the list elements on demand. Therefore we do not have to store exponentially many closed sets. Instead, we shall store just *one*!

To find the next closure we define for $A \subseteq M$ and $m_i \in M$

$$A \oplus m_i := ((A \cap \{m_1, \ldots, m_{i-1}\}) \cup \{m_i\})''.$$

We illustrate this definition by an example: Let $A := \{a, c, d, f\}$ and $m_i := e$.

$$\begin{array}{ccccccc} & & & & \downarrow & & \\ \hline \multicolumn{1}{|c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{.} \\ \hline \end{array}.$$

We first remove all elements that are greater or equal m_i from A:

$$\begin{array}{ccccccc} & & & & \downarrow & & \\ \hline \multicolumn{1}{|c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{.} \\ \hline \end{array}.$$

Then we insert m_i

$$\begin{array}{ccccccc} & & & & \downarrow & & \\ \hline \multicolumn{1}{|c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{\times} & \multicolumn{1}{c|}{.} & \multicolumn{1}{c|}{.} \\ \hline \end{array}$$

and form the closure. Since we have not yet specified the closure operator $\cdot''$ (i. e., we have not given a formal context), the example stops here with

$$A \oplus e = \{a, c, d, e\}''.$$

Proposition 13 *1. If $i \notin A$ then $A < A \oplus i$.*
2. If B is closed and $A <_i B$ then $A \oplus i \subseteq B$, in particular $A \oplus i \leq B$.
3. If B is closed and $A <_i B$ then $A <_i A \oplus i$.

Theorem 14 *The smallest closed set larger than a given set $A \subset M$ with respect to the lectic order is*

$$A \oplus i,$$

i being the largest element of M with $A <_i A \oplus i$.

Now we are ready to give the algorithm for generating all extents of a given context (G, M, I): The lectically smallest extent is $\emptyset''$. For a given set $A \subset G$ we find the lectically next extent by checking all elements i of $G \setminus A$, starting from the largest one and continuing in a descending order until for the first time $A <_i A \oplus i$. $A \oplus i$ then is the "next" extent we have been looking for. These three steps are made explicit in Figures 7 to 9.

```
Algorithm First Closure
Input:  A closure operator $X \mapsto X''$ on a finite set $M$.
Output:  The closure $A$ of the empty set.
begin
  $A := \emptyset''$;
end.
```

Fig. 7. First Closure.

```
Algorithm Next Closure
Input:  A closure operator $X \mapsto X''$ on a finite set $M$,
          and a subset $A \subseteq M$.
Output:  $A$ is replaced by the lectically next closed set.
begin
  $i :=$ largest element of $M$;
  $i :=$ succ($i$);
  $success$ := false;
  repeat
    $i :=$ pred($i$);
    if $i \notin A$ then
    begin
      $A := A \cup \{i\}$;
      $B := A''$;
      if $B \setminus A$ contains no element $< i$ then
      begin
        A:= B;
        $success$ := true;
      end;
    end else $A := A \setminus \{i\}$;
  until $erfolg$ or $i =$ smallest element of $M$.
end.
```

Fig. 8. Next Closure.

5 Implications

Have another look at the concept lattice shown in Figure 4. The six attributes describe how two unit squares can be placed with respect to each other. Each of the ten objects is a pair of unit squares, representing a possible placement. These ten pairs are representatives for an infinite set of possible positions that such pairs of squares may have. It is not stated, but perhaps expected by the reader, that these ten examples cover all possible combinations of the given attributes.

Such a situation occurs often: attributes are given, but objects are not known, or too many to handle them completely. We then have to study the possible attribute combinations, the *attribute logic* of the respective situation.

```
Algorithm All Closures
Input:  A closure operator $X \mapsto X''$ on a finite set $M$.
Output:  All closed sets in lectic order.
begin
  First_Closure;
  repeat
    Output $A$;
    Next_Closure;
  until not success;
end.
```

Fig. 9. Generating all closed sets.

Let M be some set. We shall call the elements of M *attributes*, so as if we consider a formal context (G, M, I). However we do not assume that such a context is given or explicitly known.

Definition 21 An **implication between attributes** in M is an expression of the form $A \to B$ where A and B are subsets of M. The set A is the **premise** of the implication and B is its **conclusion**.

A subset $T \subseteq M$ **respects** an implication $A \to B$ if $A \not\subseteq T$ or $B \subseteq T$. We then also say that T is a **model** of the implication $A \to B$, and denote this by $T \models A \to B$. T **respects a set** $\mathcal{L}$ of implications if T respects every single implication in $\mathcal{L}$. The implication $A \to B$ **holds** in a set $\{T_1, T_2, \ldots\}$ of subsets if each of these subsets respects $A \to B$. With

$$\mathrm{Imp}\{T_1, T_2, \ldots\}$$

we denote the set of all implications that hold in $\{T_1, T_2, \ldots\}$. $\Diamond$

5.1 Implications of a Formal Context

Now let us consider the special case of implications of a formal context.

Definition 22 $A \to B$ **holds in a context** (G, M, I) if every object intent respects $A \to B$, that is, if each object that has all the attributes in A also has all the attributes in B. We then also say that $A \to B$ *is an implication of* (G, M, I). $\Diamond$

Proposition 15 *An implication $A \to B$ holds in (G, M, I) if and only if $B \subseteq A''$, which is equivalent to $A' \subseteq B'$. It then automatically holds in the set of all concept intents as well.*

An implication $A \to B$ holds in (G, M, I) if and only if each of the implications

$$A \to m, \qquad m \in B,$$

holds ($A \to m$ is short for $A \to \{m\}$). We can read this off from a concept lattice diagram in the following manner: $A \to m$ holds if the infimum of the attribute concepts corresponding to the attributes in A is less or equal than the attribute concept for m, formally if

$$\bigwedge\{\mu a \mid a \in A\} \leq \mu m.$$

$A \to B$ holds in (G, M, I) if

$$\bigwedge\{\mu a \mid a \in A\} \leq \bigwedge\{\mu b \mid b \in B\}.$$

5.2 Semantic and Syntactic Implication Inference

As we will see, it is not necessary to store all implications of a formal context. We will discuss how implications can be derived from already known implications. First we discuss which kind of inference we want to model. This is given by the so-called *semantical inference*. Then we discuss a calculus (*syntactic inference*), and argue that the calculus is correct and complete with respect to our semantics.

5.3 When does an implication follow from other implications (semantically)?

Proposition 16 *If $\mathcal{L}$ is a set of implications in M, then*

$$\operatorname{Mod}\mathcal{L} := \{T \subseteq M \mid T \textit{ respects } \mathcal{L}\}$$

is a closure system on M. If $\mathcal{L}$ is the set of all implications of a context, then $\operatorname{Mod}\mathcal{L}$ is the system of all concept intents.

The respective closure operator

$$X \mapsto \mathcal{L}(X)$$

can be described as follows: For a set $X \subseteq M$, let

$$X^{\mathcal{L}} := X \cup \bigcup\{B \mid A \to B \in \mathcal{L}, A \subseteq X\}.$$

Form the sets $X^{\mathcal{L}}$, $X^{\mathcal{L}\mathcal{L}}$, $X^{\mathcal{L}\mathcal{L}\mathcal{L}}$, ... until[10] a set $\mathcal{L}(X) := X^{\mathcal{L}\ldots\mathcal{L}}$ is obtained with $\mathcal{L}(X)^{\mathcal{L}} = \mathcal{L}(X)$. Later on we shall discuss how to do this computation efficiently.

It is not difficult to construct, for any given set $\mathcal{L}$ of implications in M, a formal context such that $\operatorname{Mod}\mathcal{L}$ is the set of concept intents of this formal context. In fact, $(\operatorname{Mod}\mathcal{L}, M, \ni)$ will do.

Definition 23 An implication $A \to B$ **follows (semantically)** from a set $\mathcal{L}$ of implications in M if each subset of M respecting $\mathcal{L}$ also respects $A \to B$. A family of implications is called **closed** if every implication following from $\mathcal{L}$ is already contained in $\mathcal{L}$. A set $\mathcal{L}$ of implications of (G, M, I) is called **complete**, if every implication that holds in (G, M, I) follows from $\mathcal{L}$. $\diamondsuit$

In other words: An implication $A \to B$ follows semantically from $\mathcal{L}$ if it holds in every model of $\mathcal{L}$.

[10] If M is infinite, this may require infinitely many iterations.

5.4 When does an implication follow from other implications (syntactically)?

The semantic definition of implication inference has a syntactic counterpart. We can give sound and complete inference rules (known as **Armstrong rules** [1]) and an efficient algorithm for inference testing.

Proposition 17 *A set $\mathcal{L}$ of implications in M is closed if and only if the following conditions are satisfied for all $W, X, Y, Z \subseteq M$:*

1. $X \to X \in \mathcal{L}$,
2. *If* $X \to Y \in \mathcal{L}$, *then* $X \cup Z \to Y \in \mathcal{L}$,
3. *If* $X \to Y \in \mathcal{L}$ *and* $Y \cup Z \to W \in \mathcal{L}$, *then* $X \cup Z \to W \in \mathcal{L}$.

Readers with a background in Computational Logic may prefer a different notation of these Armstrong rules:

$$\frac{}{X \to X}, \qquad \frac{X \to Y}{X \cup Z \to Y}, \qquad \frac{X \to Y, \quad Y \cup Z \to W}{X \cup Z \to W}.$$

The proposition says that a set of implications is the set of all implications of some context if and only if it is closed with respect to these rules. In other words, an implication follows from other implications if and only if it can be derived from these by successive applications of these rules. In particular, semantic and syntactic inference are the same.

However, these rules do not always suggest the best proof strategy. Instead, we may note the following:

Proposition 18 *An implication $X \to Y$ follows from a list $\mathcal{L}$ of implications if and only if $Y \subseteq \mathcal{L}(X)$.*

We give an algorithm that efficiently computes the closure $\mathcal{L}(X)$ for any given set X. Such algorithms are used in the theory of relational data bases for the study of functional dependencies.

We can give a rough complexity estimation of the algorithm in Figure 10. Except for manipulations of addresses, the main effort is to apply the implications. Each implication is applied at most once, and each application requires a simple set operation. Therefore the time required by the closure algorithm is essentially *linear* in the size of the input $\mathcal{L}$.

Summarizing these considerations we learn that implication inference is easy: to check if an implication $X \to Y$ follows from a list $\mathcal{L}$ of implications, it suffices to check if $Y \subseteq \mathcal{L}(X)$ (by Proposition 18), and this can be done in time linear in the size of the input.

In other words: implications are easy to use, much easier than many other logical constructs. This may be a reason why implications are popular, and perhaps be part of an explanation why our simple theory of formal concepts is so useful.

```
Algorithm CLOSURE
Input:  A list $\mathcal{L} =: [\mathcal{L}[1], \dots, \mathcal{L}[n]]$ of implications in $M$
          and a set $X \subseteq M$.
Output:  The closure $\mathcal{L}(X)$ of $X$.
begin
  for all $x \in M$ do
  begin
    avoid[$x$] := $\{1, \dots, n\}$;
    for $i := 1$ to $n$ do with $A \to B := \mathcal{L}[i]$
      if $x \in A$ then avoid[$x$] :=avoid[$x$] $\setminus \{i\}$;
  end;
  used_imps :=$\emptyset$;
  old_closure :=$\{-1\}$;     (* some element not in $M$ *);
  new_closure := $X$;
  while new_closure $\neq$ old_closure do
  begin
    old_closure := new_closure;
    $T := M \setminus$new_closure;
    usable_imps := $\bigcap_{x \in T}$avoid[$x$];
    use_now_imps := usable_imps $\setminus$ used_imps;
    used_imps := usable_imps;
    for all $i \in$ use_now_imps with $A \to B := \mathcal{L}[i]$ do
      new_closure := new_closure $\cup B$;
  end;
  $\mathcal{L}(X)$ :=new_closure;
end.
```

Fig. 10. Algorithm CLOSURE.

5.5 The Stem Base

The number of implications that hold in a given situation can be very large. For example, if there is only one closed set, M, then *every* implication holds. If M has n elements, then these are some 2^{2n} implications. But this is ridiculous, because all these implications can be inferred from a single one, namely from $\emptyset \to M$.

We see from this trivial example that the set of *all* implications of a given formal context may be highly redundant. It is a natural question to ask for a small *implicational base*, from which everything else follows. More precisely we ask, for any given formal context (G, M, I), for a list $\mathcal{L}$ of implications that is

– sound (i.e., each implication in $\mathcal{L}$ holds in (G, M, I)),
– complete (i.e., each implication that holds in (G, M, I) follows from $\mathcal{L}$), and
– non-redundant (i.e., no implication in $\mathcal{L}$ follows from other implications in $\mathcal{L}$).

It is easy to see that (for finite M) such sets $\mathcal{L}$ exist. We may start with some sound and complete set of implications, for example, with the set of all implications that hold in (G, M, I). We then can successively remove redundant implications from this set, until we obtain a sound, complete, non redundant set.

But this is an unrealistic procedure. We therefore look for a better way to construct an implicational base. Duquenne and Guigues [5] have shown that there is a natural choice, the *stem base*.

The following recursive definition is rather irritating at the first glance. We define a pseudo-closed set to be a set which is not closed, but contains the closure of every pseudo-closed proper subset.

Definition 24 Let $X \mapsto X''$ be a closure operator on the finite set M. We call a subset $P \subseteq M$ **pseudo-closed**, if (and only if)

- $P \neq P''$, and
- if $Q \subset P$ is a pseudo-closed proper subset of P, then $Q'' \subseteq P$. $\Diamond$

This *is* a valid recursive definition. It is not circular, because the pseudo-closed set Q must have fewer elements than P, because it is a proper subset.[11]

Reformulating this definition for formal contexts, we obtain the following definition.

Definition 25 Let (G, M, I) be a formal context with M finite. A subset $P \subseteq M$ is a **pseudo intent** of (G, M, I) iff

- P is not a concept intent, and
- if $Q \subset P$ is a pseudo-closed proper subset of P, then there is some object $g \in G$ such that $Q \subseteq g'$ but $P \not\subseteq g'$. $\Diamond$

Theorem 19 *Let M be finite and let $X \mapsto X''$ be some closure operator on M. The set of implications*

$$\{P \to P'' \mid P \textit{ pseudo-closed } \}$$

is sound, complete, and non redundant.

The implication set in the theorem deserves a name. It is sometimes called the *Duquenne–Guigues–base*. We simply call it the **stem base** of the closure operator (or the *stem base of a given formal context*, if the closure operator is given that way). In practice one uses a slightly different version of the stem base, namely

$$\{P \to P'' \setminus P \mid P \text{ pseudo-closed } \}.$$

The stem base is not the only implicational base, but it plays a special rôle. For example, no implicational base can consist of fewer implications, as the next proposition shows:

Proposition 20 *Every sound and complete set of implications contains, for every pseudo closed set P, an implication $A \to B$ with $A'' = P''$.*

[11] 'Recursive' is meant here with respect to set inclusion. Compare with the following recursive definition: A natural number is *prime* iff it is greater than 1 and not divisible by any smaller prime number.

5.6 Computing the Stem Base

As before, consider a closure operator $X \mapsto X''$ on a finite set M. We start with a harmless definition:

Definition 26 A set $Q \subseteq M$ is **$\bullet$-closed** if it contains the closure of every $\bullet$-closed set that is properly contained in Q.

Formally, Q is $\bullet$-closed iff for each $\bullet$-closed set $Q_0 \subset Q$ with $Q_0 \neq Q$ we have $Q_0'' \subseteq Q$. $\Diamond$

This is a simple (but convenient) renaming, as the next proposition shows.

Proposition 21 *A set is $\bullet$-closed iff it is either closed or pseudo-closed.*

Observe that if Q contains the closure of every $\bullet$-closed subset, then Q must be closed.

The first crucial step towards finding pseudo-closed sets is this:

Proposition 22 *The intersection of $\bullet$-closed sets is $\bullet$-closed.*

In other words: the $\bullet$-closed sets form a closure system. We have described an algorithm to compute, for a given closure operator, all closed sets. We can apply this algorithm for computing all $\bullet$-closed sets, provided that we can access the corresponding closure operator. This is easy. We prepare the result with a proposition that is an immediate consequence of Definition 26.

Proposition 23 *Q is $\bullet$-closed iff Q satisfies the following condition:*

If $P \subset Q$, $P \neq Q$, is pseudo-closed, then $P'' \subseteq Q$.

Proposition 23 shows how to find the quasi closure of an arbitrary set $S \subseteq M$: As long as the condition in the proposition is violated, we (are forced to) extend the set S, until we finally reach a fixed point.

Let $\mathcal{L}$ be the stem base[12]. Define, for $X \subseteq M$,

$$X^{\mathcal{L}^\bullet} := X \cup \bigcup \{P'' \mid P \to P'' \in \mathcal{L}, P \subset X, P \neq X\},$$

iterate by forming

$$X^{\mathcal{L}^\bullet \mathcal{L}^\bullet}, X^{\mathcal{L}^\bullet \mathcal{L}^\bullet \mathcal{L}^\bullet}, \ldots$$

until a set

$$\mathcal{L}^\bullet(X) := X^{\mathcal{L}^\bullet \mathcal{L}^\bullet \ldots \mathcal{L}^\bullet}$$

is obtained that satisfies

$$\mathcal{L}^\bullet(X) = \mathcal{L}^\bullet(X)^{\mathcal{L}^\bullet}.$$

Proposition 24 *$\mathcal{L}^\bullet(X)$ is the smallest $\bullet$-closed set containing X.*

```
Algorithm $\mathcal{L}^\bullet$–Closure
Input:  A list $\mathcal{L} =: [\mathcal{L}[1], \ldots, \mathcal{L}[n]]$ of implications in $M$
          and a set $X \subseteq M$.
Output:  The closure $\mathcal{L}(X)$ of $X$.
begin
  for all $x \in M$ do
  begin
    avoid[$x$] := $\{1, \ldots, n\}$;
    for $i := 1$ to $n$ do with $A \to B := \mathcal{L}[i]$
      if $x \in A$ then avoid[$x$] :=avoid[$x$] $\setminus \{i\}$;
  end;
  used_imps :=$\emptyset$;
  old_closure :=$\{-1\}$;     (* some element not in $M$ *);
  new_closure := $X$;
  while new_closure $\neq$ old_closure do
  begin
    old_closure := new_closure;
    $T := M \setminus$ new_closure;
    usable_imps := $\bigcap_{x \in T}$ avoid[$x$] $\cap \bigcup_{x \in \mathit{new\_closure}}$ avoid[$x$];
    use_now_imps := usable_imps $\setminus$ used_imps;
    used_imps := usable_imps;
    for all $i \in$ use_now_imps with $A \to B := \mathcal{L}[i]$ do
      new_closure := new_closure $\cup B$;
  end;
  $\mathcal{L}(x)$ :=new_closure;
end.
```

Fig. 11. Computing the $\mathcal{L}^\bullet$–Closure.

Note that in order to find the quasi closure, we use only pseudo-closed sets which are contained in the closure and therefore in particular are lectically smaller than the quasi closure. Thus, the same result is obtained if $\mathcal{L}$ is a subset of the stem base, containing thoses implications $P \to P''$ for which P is pseudo-closed and lectically smaller than $\mathcal{L}^\bullet(X)$.

Now it is easy to give an algorithm to compute all pseudo-closed sets for a given closure operator. We use the Next Closure algorithm applied to the closure system of $\bullet$-closed sets. For short, we shall refer to this as the **next quasi closure** after a given set A, next_$\mathcal{L}^\bullet$_closure(A). This produces all $\bullet$-closed sets in lectic order. We record only those which are not closed. This yields a list of all pseudo-closed sets.

Since the $\bullet$-closed sets are generated in lectic order, we have, at each step, the full information about the lectically smaller pseudo-closed sets. We have seen that this suffices to compute the "quasi closure" operator. The algorithm

[12] The reader might wonder why we use the stem base to construct the stem base. As we shall see soon, this works, due to the recursive definition of the stem base.

in Figure 12 uses a dynamic list $\mathcal{L}$. Whenever a pseudo-closed set P is found, the corresponding implication $P \to P''$ is included in the list. Since the pseudo-closed sets are found in lectic order, this makes sure that at any step we have sufficient information to compute the quasi closure.

```
Algorithm Stem base
Input:   A closure operator $X \mapsto X''$ on a finite set $M$,
           for example given by a formal context $(G, M, I)$.
Output:  The stem base $\mathcal{L}$

begin
  $\mathcal{L} := \emptyset$;
  $A := \emptyset$;
  while $A \neq M$ do
  begin
    if $A \neq A''$ then $\mathcal{L} := \mathcal{L} \cup \{A \to A''\}$;
    $A :=$ next_$\mathcal{L}^{\bullet}$_closure($A$);
  end;
end.
```

Fig. 12. Computing the stem base for a given closure operator.

Example 2 We compute the stem base for the context of triangles given in Example 1. The steps are shown in Fig. 13. The first column contains all quasi closed sets, in lectic order. The pseudo-closed sets are precisely those which are not closed (see middle column). Each pseudo-closed set gives rise to an entry in the stem base (last column, short form).

Since the closure operator is given in terms of a formal context, we may speak of **quasi intent**s and **pseudo intent**s instead of $\bullet$-closed sets or pseudo-closed sets. We see that the algorithm generates all quasi intents to find the stem base. In other words, to compute all pseudo intents we also compute all intents, possibly exponentially many. This looks like a rather unefficient method. Unfortunately, we do not know of a better strategy. It is an open problem to find a better algorithm for the stem base. In practice, the algorithm is not fast, but nevertheless very useful.

6 Conclusion

Much more can be said about FCA, here we only dealt with the foundations of the discipline. Over the past decades, the field has expanded in many directions. We give a few examples of central topics in FCA which weren't discussed here.

$\bullet$-closed set	closed ?	stem base implication
$\emptyset$	yes	
$\{e\}$	yes	
$\{d\}$	yes	
$\{d, e\}$	no	$\{d, e\} \to \{a, b, c\}$
$\{c\}$	yes	
$\{c, e\}$	no	$\{c, e\} \to \{a, b, d\}$
$\{c, d\}$	no	$\{c, d\} \to \{a, b, e\}$
$\{b\}$	yes	
$\{b, e\}$	yes	
$\{b, d\}$	yes	
$\{b, c\}$	yes	
$\{a\}$	no	$\{a\} \to \{b, c\}$
$\{a, b, c\}$	yes	
$\{a, b, c, d, e\}$	yes	

Fig. 13. Steps in the stem base algorithm

– **Conceptual Scaling.** One can argue that formal contexts are only able to represent very limited information (essentially yes/no). The idea to allow for proper entries in the tables rather than just crosses or blanks leads to the notion of *multi-valued contexts* which are closer to database tables typically encountered in practice. A very generic method to make the machinery of FCA applicable to data represented as multi-valued contexts is called *conceptual scaling*, where this data is transformed back into plain formal contexts.
– **Association Rules.** Real world data is often noisy and error-prone. In order to still extract meaningful implicational information from such data, one needs to formalize the notion of implications which do not hold always (but often). *Association rules* are such implications that come with two values, *support* (the fraction of all objects where the implication is applicable and valid) and *confidence* (the fraction of objects satisfying the implication among all objects where it is applicable). For association rules, one can again characterize semantic and syntactic consequences and establish implicational bases [7].
– **Triadic FCA.** The readers might ask themselves, why the incidence relation used to characterize data in FCA is a binary one. Couldn't it have a higher arity? Indeed, people have investigated *triadic FCA* [6], where the incidence relation is a ternary one between objects, attributes and conditions. Some of the notions of FCA can be nicely generalized to the triadic case (and even to incidence relations of higher arity, giving rise to *polyadic FCA* [8]) but others are specific to the binary case.
– **Attribute Exploration.** Sometimes the data of a domain is only partially recorded in a formal context, but there are experts who know the full domain of interest. In that case, algorithms exist which can complete the context and

determine all the implications in an interactive process, where an expert is repeatedly asked questions about the domain [3].

Acknowledgments.

We are grateful for the valuable feedback from the anonymous reviewers, which helped greatly to improve this work. Special thanks to Thomas Feller for his very careful proof-reading. This work has been funded by the European Research Council via the ERC Consolidator Grant No. 771779 (DeciGUT)

References

1. Armstrong, W.: Dependency structures of data base relationships. In: Proc. of IFIP Congress. pp. 580–583 (1974)
2. Arnauld, A., Nicole, P.: La logique ou l'art de penser — contenant, outre les règles communes, plusieurs observations nouvelles, propres à former le jugement. Ch. Saveux, Paris (1668)
3. Ganter, B., Obiedkov, S.A.: Conceptual Exploration. Springer (2016)
4. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1997)
5. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. Mathématiques et Sciences Humaines **95**, 5–18 (1986)
6. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: Ellis, G., Levinson, R., Rich, W., Sowa, J.F. (eds.) Proc. 3rd Int. Conf. on Conceptual Structures (ICCS 1995). LNCS, vol. 954, pp. 32–43. Springer (1995)
7. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques, Informatique et Sciences Humaines (29), 35–55 (1991)
8. Voutsadakis, G.: Polyadic concept analysis. Order - A Journal on The Theory of Ordered Sets and Its Applications **19**(3), 295–304 (2002)