Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 69 (2014) 1258 – 1267

Procedia Engineering

www.elsevier.com/locate/procedia

24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013

Formal Concept Analysis – Overview and Applications

Frano Škopljanac-Mačina*, Bruno Blašković

University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb 10000, Croatia

Abstract

In this article we give a brief overview of the theory behind the formal concept analysis, a novel method for data representation and analysis. From given tabular input data this method finds all formal concepts and computes a concept lattice, a directed, acyclic graph, in which all formal concepts are hierarchically ordered. We describe the link between this method and formal logic, as well as graph theory. Finally we present one example of an application of this method in the field of computer aided learning.

Keywords: formal data analysis; FCA; lattice; line diagram; concept hierarchy; computer aided learning

1. Introduction

When dealing with large sets of data we inevitably need to address the problem of data representation. Many different approaches for tackling that problem have been proposed and applied over the years, e.g. various tree structures, dendrograms and, more recently, concept lattices. In this paper we will focus on concept lattices and their underlying theory of Formal concept analysis. We propose a simple and effective way of applying Formal concept analysis in the area of teaching and computer aided learning. Specifically, we want to help instructors with the preparation of exam materials, so they can check how a given set of questions covers specific concepts. A given set of questions represents the input data for Formal concept analysis, and the resulting concept lattice provides teachers with a visual overview of the proposed exam structure, and key information about the found concepts. Furthermore, the same concept lattice can be used for analyzing exam results to identify which concepts are harder for students to learn and understand. Also, we propose to use exam's concept lattice to optimize the order of questions, by applying

* Corresponding author. Tel.: +38516129763.
E-mail address: frano.skopljanac-macina@fer.hr

doi:10.1016/j.proeng.2014.03.117

topological sorting algorithms to the concept lattice. Finally, we discuss the possibility of using a concept lattices as a basis for building ontologies.

Formal concept analysis (FCA, originally in German *Formale Begriffsanalyse*) is a method for knowledge representation, information management and data analysis. This method finds and visualizes all concepts and their dependencies from the tabular input data. Formal concept analysis has been applied in various fields such as mathematics, medicine, biology, sociology, psychology or economics. Most interesting applications of FCA are arguably in computer science – it can be used for data mining, data analysis, information retrieval, source code error correction, machine learning and for building taxonomies and ontologies [5,6,7,8].

2. Formal concept analysis

FCA method was devised in the early 1980s by R. Wille, a German mathematician and professor emeritus at the TU Darmstadt. He used the philosophical interpretation of the concept as a unit of thought comprising of a set of objects and a set of their shared attributes. Theoretical foundations of FCA are built on applied lattice theory and set theory [1,2,3,4,5].

Input data for FCA method is represented in matrix form, so that each row represents an object from the domain of interest, and each column represents one of the defined attributes. Input matrix elements can only assume Boolean values, i.e. any object either has or does not have a particular attribute. If an object has a particular attribute a mark (e.g. symbol *"X"*) is placed on the intersection of that object's row and that attribute's column. Otherwise, if an object does not have a certain attribute the intersection of that object's row and that attribute's column is left blank. This input matrix is defined as a *formal context* on which the analysis will be performed. FCA method results in two sets of output data. The first set gives a hierarchical relationship of all the established concepts in the form of line diagram called *a concept lattice* (originally in Germ. *Bergiffsverband*). The second set gives a list of all found interdependencies among attributes in the formal context.

Definition 1: Formal context in FCA method is a triple $\langle X,Y,I\rangle$ where X and Y are non-empty sets and I is a binary relation between X and Y.

The formal context $\langle X,Y,I\rangle$ of an input matrix of n rows and m columns consists of a set of objects defined as $X=\{x_1,...,x_n\}$, a set of attributes defined as $Y=\{y_1,...,y_m\}$ and a binary relation I defined as $\langle x_i,y_j\rangle \in I$ if and only if the intersection of i-th row and j-th column is not blank, i.e. object x_i has an attribute y_j.

Definition 2: For a formal context $\langle X,Y,I\rangle$ we define concept-forming operators $^{\uparrow}$: $2^X \rightarrow 2^Y$ and $^{\downarrow}$: $2^Y \rightarrow 2^X$ for each $A \subseteq X$ and $B \subseteq Y$ as: $A^{\uparrow}=\{y \in Y \mid \text{for each } x \in A: \langle x,y\rangle \in I\}$ and $B^{\downarrow}=\{x \in X \mid \text{for each } y \in B: \langle x,y\rangle \in I\}$.

$A^{\uparrow}$ represents a set of all attributes shared by all objects form set A, and analogously $B^{\downarrow}$ represents a set of all objects which share all the attributes from set B. This leads to the notion of a *formal concept* as a segment of the formal context in which different objects share the same attributes.

Definition 3: Formal concept in a formal context $\langle X,Y,I\rangle$ is a pair $\langle A,B\rangle$, where $A \subseteq X$ and $B \subseteq Y$, for which $A^{\uparrow}=B$ and $B^{\downarrow}=A$.

Therefore, $\langle A,B\rangle$ is a formal concept if and only if set A consists only of those objects which have all attributes from set B (extent of the concept), and set B consists only of those attributes which are shared by all objects from the set A (intent of the concept). Concept's extent and intent can be formally defined using concept-forming operators $^{\downarrow}$ and $^{\uparrow}$, i.e. concepts' extent as $\mathrm{Ext}(X,Y,I)=\{B^{\downarrow} \mid B \in Y\}$ and intent as $\mathrm{Int}(X,Y,I)=\{A^{\uparrow} \mid A \in X\}$.

We can also define formal concepts visually within the formal context as the maximal rectangles containing only symbols *"X"* and no blank places.

It is important to note that the formal concepts are hierarchically organized with the subconcept–superconcept relation.

Definition 4: For formal concepts $\langle A_1,B_1\rangle$ and $\langle A_2,B_2\rangle$ in a formal context $\langle X,Y,I\rangle$ it holds that $\langle A_1,B_1\rangle \leq \langle A_2,B_2\rangle$ if and only if $A_1 \subseteq A_2$ and $B_2 \subseteq B_1$.

Relation $\leq$ is defined as a partial order between concepts, and it formally describes their subconcept–superconcept relationship, analogous to object oriented subclass-superclass principle. All the formal concepts from a formal context along with all their relationships can be transformed into a concept lattice structure which can be presented as a Hesse (line) diagram [16].

Definition 5: Concept lattice of a formal context $\langle X,Y,I\rangle$ is a structure defined as $\langle \boldsymbol{B}(X,Y,I),\leq\rangle$, where $\boldsymbol{B}(X,Y,I)$ is a collection of all formal concepts, $\boldsymbol{B}(X,Y,I)=\{\langle A,B\rangle\in 2^X\times 2^Y \mid A^{\uparrow}=B, B^{\downarrow}=A\}$ and $\leq$ is a partial order relation.

Concept lattice structure satisfies the mathematical definition of a lattice because it is a partially ordered set whose any two elements (formal concepts) have a supremum (*join* – mutual superconcept) and an infimum (*meet* – mutual subconcept) [16,17].

Example: On the left of the Fig. 1 there is a formal context containing 10 objects (freshmen students) and 5 attributes (first semester freshman year courses at FER). In the formal context all the passed courses are marked (*"X"*) for each student. On the right of the Fig. 1 there is a concept lattice generated from the formal context on the left. We used *Lattice Miner Platform 1.4* program for generating concept lattice. We chose a reduced display, i.e. attributes are displayed only in the first formal concept occurrence (colored in blue), along with just the number of objects for each formal concept instead of an object list (colored in red).

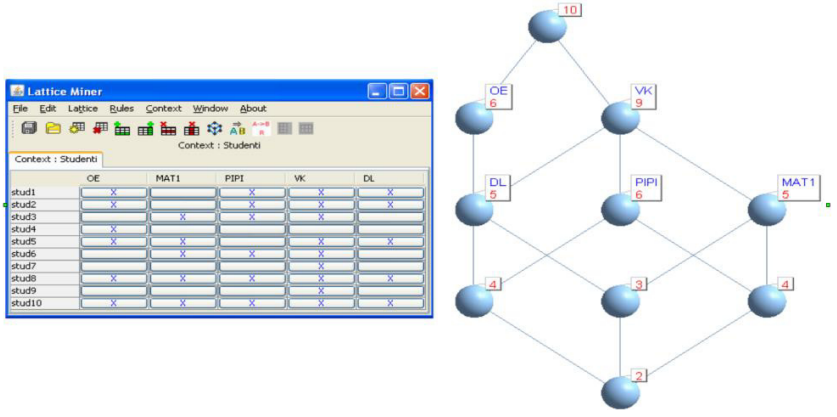

Fig. 1. example of a simple formal context (on the left) and its concept lattice (on the right)

FCA method is mathematically strictly defined, and it is based on Galois connections and closure operators [17]. Concept forming operators $^{\uparrow}: 2^X\to 2^Y$ and $^{\downarrow}: 2^Y\to 2^X$, along with partial order relation $\leq$ form a Galois connection (direct link between formal context's attribute set and object set). We obtain closure operators by composing the concept forming operators, $^{\uparrow\downarrow}: 2^X\to 2^X$ and $^{\downarrow\uparrow}: 2^Y\to 2^Y$.

The core theorem behind FCA method states that the concept lattice is a complete lattice – a partially ordered set in which *all* subsets have a supremum (*superconcept*) and infimum (*subconcept*) [16,17]. We will give a formal statement of this theorem, and the proof can be found in [4,5].

Theorem 1 (Central concept lattice theorem – R. Wille, 1982): $\boldsymbol{B}(X,Y,I)$ is a complete lattice whose infima and suprema are given by the following expressions respectively:

$$\bigwedge_{j\in J}\langle A_j,B_j\rangle=\left\langle \bigcap_{j\in J}A_j,\left(\bigcup_{j\in J}B_j\right)^{\downarrow\uparrow}\right\rangle \text{ and } \bigvee_{j\in J}\langle A_j,B_j\rangle=\left\langle \left(\bigcup_{j\in J}A_j\right)^{\uparrow\downarrow},\bigcap_{j\in J}B_j,\right\rangle.$$

Furthermore, any complete lattice $\mathbf{V}=(V,\leq)$ is isomorphic to a concept lattice $\boldsymbol{B}(X,Y,I)$ if and only there are mappings $\gamma: X\to V$ and $\mu: Y\to V$, such that $\gamma(X)$ is $\vee$-dense in V, $\mu(Y)$ is $\wedge$-dense in V, and $\gamma(X)\leq\mu(Y)$ if and only if $\langle x,y\rangle\in I$.

Possibly there can be multiple identical rows or columns in the formal context. In that case we can clean up (*clarify*) the formal context by removing redundant rows or columns. Concept lattices obtained from the clarified formal context and from the original formal context are isomorphic (there is a bijective mapping between the elements of the two lattices). Also, there are rules for reducing the formal context by safely removing specific rows

or columns, which can be especially useful when dealing with very large formal contexts. Again, concept lattices obtained from the reduced formal context and the original formal context are isomorphic [5].

This is one of the simplest algorithms for generating concept lattices:

```
1. compute extension Ext(X,Y,I)={B↓ | B∈Y}
2. for ∀A∈Ext(X,Y,I) find ⟨A,A↑⟩
```

There are also various more advanced and efficient algorithms for computing concept lattices. Less complex algorithms take the formal context $\langle X,Y,I\rangle$ and generate only the structure $\boldsymbol{B}(X,Y,I)$ (list of all formal concepts) as an output. From this structure we can infer the hierarchy between the formal concepts and obtain the concept lattice. Complex algorithms are more efficient because they directly produce the concept lattice $\langle \boldsymbol{B}(X,Y,I),\leq\rangle$ as an output.

B. Ganter has published in 1987 `NextClosure` algorithm [10], the basic algorithm for computing of structure $\boldsymbol{B}(X,Y,I)$. As an input algorithm takes the formal context $\langle X,Y,I\rangle$, and as an output algorithm produces $\mathrm{Int}(X,Y,I)$ – lexicographically ordered list of all intensions from the formal context. Structure $\boldsymbol{B}(X,Y,I)$ can be reconstructed from that list because $\boldsymbol{B}(X,Y,I)=\{\langle B^{\downarrow},B\rangle \mid B\in\mathrm{Int}(X,Y,I)\}$. Finally, by applying Definition 4 we can establish all the hierarchical relations from the structure $\boldsymbol{B}(X,Y,I)$, which is necessary for generating the concept lattice.

The Ganter's algorithm uses lexicographic successor theorem, so first we will give necessary definitions and formally state that theorem (its proof can be found in [10,5]).

Definition 6: Attributes $A,B\subseteq Y$, $i\in\{1,...,n\}$ are lexicographically ordered ($A <_i B$) if and only if: $A\cap\{1,...,i\text{-}1\}=B\cap\{1,...,i\text{-}1\}$ where $i\in B-A$.

Definition 7: We define operation $A\oplus i:=((A\cap\{1,...,i-1\})\cup\{i\})^{\downarrow\uparrow}$ for $A\subseteq Y$, $i\in\{1,...,n\}$.

Theorem 2 (lexicographical successor):

The least intension B^{+} larger than $B\subseteq A$ is given by $B^{+}=B\oplus i$ where i is the largest element of Y for which $B <_i B\oplus i$ holds.

Now we can give the pseudo code of the `NextClosure` algorithm:

```
1. A:= ∅↓↑; (least intention)
2. store (A);
3. while not (A=Y) do
4.    A:=A+; (lex. successor)
5.    store(A);
6. endwhile.
```

The time complexity of the `NextClosure` algorithm using the big O notation is: $O(|X|\cdot|Y|^2\cdot|\boldsymbol{B}(X,Y,I)|)$.

In the Fig. 1 we presented a case of a simple concept lattice from a formal context with a smaller number of objects and attributes. The size of the concept lattice can grow exponentially with the increase of the formal context, and for larger input data sets it can become very dense and illegible. Therefore, we can break up a formal context in sections by grouping attributes in smaller sets. After that we can compute concept lattices for every section of the formal context separately. It is possible to compute a product of these smaller concept lattices, which gives us a clearer overview of the whole concept lattice. Big concept lattices can be simplified by using concept clustering techniques. In that case concept lattice shows only those formal concepts which have sufficient support, i.e. greater than a given minimum value from a [0,1] interval. Support is calculated similarly to a support measure from association rules in data mining: $\mathrm{supp}(B\subseteq Y)=|B^{\downarrow}|/|X|$, i.e. support of a set of attributes is a number of all objects with those attributes divided by the total number of objects in the formal context. Therefore we can reduce and simplify concept lattice by increasing the minimal support.

From the concept lattice or the formal context itself we can explore attribute interdependencies or attribute implications – e.g. *"integer divisible by 3 and 4 is also divisible by 1, 2, 6 and 12"*. Attribute implications can be directly read from the concept lattice, because every formal concept necessarily contains all the attributes from all its superconcepts. Nevertheless we can formally define attribute implications as a link between the FCA method and formal logic. Determining the attribute implications is especially useful when the formal context is very large and

the concept lattice is dense and hard to read. Then we can find a minimum set of all attribute implications which are true in that formal context.

Definition 8: Attribute implication over a non-empty attribute set Y is defined by expression $A \Rightarrow B$ where $A \subseteq Y$ and $B \subseteq Y$.

Definition 9: Attribute implication $A \Rightarrow B$ over set Y is valid (true) in a set $M \subseteq Y$ if and only if implication $A \subseteq M \Rightarrow B \subseteq M$ holds.

Let set M consist of only one row in the formal context (e.g. set of all attributes of an object x: $x^{\uparrow}$). Then we check the validity of attribute implication $A \Rightarrow B$, i.e. if an object has all attributes from A then it has all attributes from B. If this implication is true then we write $\|A \Rightarrow B\|_M = 1$, and $\|A \Rightarrow B\|_M = 0$ otherwise. It is interesting to determine the validity of $\|A \Rightarrow B\|_{\langle X,Y,I \rangle}$, attribute implication over the entire formal context. This implication is true, $\|A \Rightarrow B\|_{\langle X,Y,I \rangle} = 1$, if and only if for every row (object) of the formal context $\|A \Rightarrow B\|_M = 1$. Furthermore $A \Rightarrow B$ is true in the formal context if and only if $A^{\downarrow} \subseteq B^{\downarrow}$ and $B \subseteq A^{\downarrow\uparrow}$ holds. Here we must discuss one problem – the possibility of finding redundant attribute implications. One way of solving that problem is to determine *the theory* and *the model* of the formal context. The theory contains a set of attribute implications over the formal context. Theory **T** can be made nonredundant, so that it does not contain superfluous implications which just follow from other basic implications. Theory model **M** represents an attribute subset M of a formal context in which every implication from the theory **T** is true. The set of all these models of theory **T** is denoted by Mod(**T**) and it is formally defined as: $\text{Mod}(\mathbf{T}) = \{M \subseteq Y \mid \forall A \Rightarrow B \in \mathbf{T}: \|A \Rightarrow B\|_M = 1\}$.

Definition 10: Attribute implication $A \Rightarrow B$ semantically follows from the theory ($\mathbf{T} \models A \Rightarrow B$) if and only if $A \Rightarrow B$ is true in every model **M** of theory **T**.

Therefore if we want to check if an attribute implication semantically follows from a theory **T**, we must find Mod(**T**) (set of all models of theory **T**) and for its every element check if the implication is true.

We can also, by applying simple deduction rules, directly infer new attribute implications over the theory **T**, without the need to check their validity in the set Mod(**T**). To accomplish this FCA method uses Armstrong's rules which were introduced in the relational database designing for determining functional dependencies [13,5]. There are two basic inference rules:

$$\frac{}{A \bigcup B \Rightarrow A} \text{ and } \frac{A \Rightarrow B, B \bigcup C \Rightarrow D}{A \bigcup C \Rightarrow D}.$$

But we can also use these useful derived rules:

$$\frac{}{A \Rightarrow A}, \frac{A \Rightarrow B}{A \bigcup C \Rightarrow B}, \frac{A \Rightarrow B, A \Rightarrow C}{A \Rightarrow B \bigcup C}, \frac{A \Rightarrow B \bigcup C}{A \Rightarrow B} \text{ and } \frac{A \Rightarrow B, B \Rightarrow C}{A \Rightarrow C}.$$

Proof of a new attribute implication $A \Rightarrow B$ from theory **T** is written as a sequence of implications which are obtained by successively using these deduction rules. This way we can show that $A \Rightarrow B$ syntactically follows from the theory **T** ($\mathbf{T} \vdash A \Rightarrow B$). As these specified deduction rules are sound and syntactically and semantically complete, we must note that the following important equivalence is also true: $\mathbf{T} \vdash A \Rightarrow B \Leftrightarrow \mathbf{T} \models A \Rightarrow B$. Theory **T** is syntactically closed if it contains all attribute implication which can be derived from **T**. Furthermore, if the theory **T** is syntactically closed then it is also semantically closed (and vice versa). After the syntactically (and semantically) closed set of theory **T** of a formal context $\langle X,Y,I \rangle$ is found it can be shown that the set of all theory models is identical to the set of all intentions of a concept lattice: $\text{Mod}(\mathbf{T}) = \text{Int}(X,Y,I)$.

There is also a more efficient way of checking if an attribute implication semantically follows for a theory **T**. Instead of checking the validity of implication $A \Rightarrow B$ for every model of a theory **T**, we can check if it is true only for $C_{\text{Mod}(\mathbf{T})}(A)$ – the smallest model of theory **T** which contains attribute set A (antecedent of implication $A \Rightarrow B$). If the whole set of all models of theory **T** is not known, we can compute, using various known algorithms, only that smallest model, which we will use for checking if $\|A \Rightarrow B\|_{C_{\text{Mod}(\mathbf{T})}(A)} = 1$.

As we already mentioned we can ensure that the theory **T** is nonredundant, namely that there are no redundant implications which can be inferred using deduction rules. Theory **T** of a formal context $\langle X,Y,I\rangle$ is *complete* if and only if its every valid attribute implication follows from theory **T**. Also the theory **T** is complete if $\mathrm{Mod}(\mathbf{T})=\mathrm{Int}(X,Y,I)$ is true. Theory **T** is nonredundant if it is complete, and if after removing any implication it would not semantically follow from theory **T**. Algorithm for removing redundant implications checks if an implication is still a semantic consequence of theory **T** after we delete it from the theory **T**. If the condition is met, the algorithm continues until no such implication is found, i.e. until the theory **T** becomes nonredundant.

Researchers have devised multiple complex and advanced algorithms for the computation of nonredundant basis of formal context theory, such as Guigues and Duquenne's algorithm [9,11].

3. Topological sorting

Concept lattice with the partial order $\langle \boldsymbol{B}(X,Y,I),\leq\rangle$ is a directed acyclic graph (DAG) whose nodes are formal concepts connected by directed edges (always from superconcepts to subconcepts). Therefore, we can analyze concept lattices using various algorithms and techniques from the field of graph theory. It can be useful to find a linear sorted list of formal concepts, with regard to all subconcept-superconcept relations between them. The task can be solved using known algorithms for topological sorting. These algorithms provide a linear sorted list in a way that for every edge uv of the directed acyclic graph G node u is added to the sorted list before node v. That means that every node enters linear sorted list only after all its parent nodes are in the list or if it doesn't have any parent nodes. Stated more formally, topological sorting algorithms provide linearly ordered set which corresponds to a partial ordered set (e.g. concept lattice).

There are numerous topological sorting algorithms, but they can all be divided into two groups – the first group contains algorithms based on source extracting (nodes with no incoming edges), and the other group consists of depth first search (DFS) based algorithms. We will describe the basic principles behind both types of algorithms.

Source extracting algorithms start by looking for node(s) with no incoming edges, erase them (along with their outgoing edges) and add them to the list. These two steps are repeated until the graph is empty and the linear sorted list of nodes is full [12,14].

Depth first search based algorithms use DFS method for traversing the graph (exploring graph from the starting node along every edge before backtracking) and notes the order of fully explored nodes. Finally, by reversing the list of fully explored nodes we obtain a linear sorted list of nodes [12,14].

Implementations of these algorithms can be more complex, e.g. they also check for cycles, and if there are cycles the sorting ends immediately, because directed cyclic graphs (DCG) cannot be topologically sorted. The time complexity of these algorithms is $O(|V|+|E|)$ in big O notation, meaning it is linearly dependent on the number of nodes and edges.

It is possible to get multiple different correct final orders of nodes - the solution is not unique. But if a graph has a directed Hamiltonian path (spanning path - visits every node exactly once) every topological sorting algorithm will yield an identical solution [15]. For example, a chain-like concept lattice has a Hamiltonian path, but then the order of formal concepts can be directly read from such concept lattice without using special algorithms.

4. Example of FCA method application

It was stated in the introduction that the FCA method can be used in various areas. In this example we discuss the possibility of using the FCA method in the field of teaching as a support system for designing exams, and subsequent analysis of exam results, as well as a useful addition to the e-learning systems.

Here we focus on a small set of exam tasks, and we try to find their similarities, differences and mutual relations. We have picked 10 exam tasks from the Fundamentals of electrical engineering, a class from the freshman year of the Bachelor program in the Faculty of electrical engineering and computing, University of Zagreb, in the academic year 2010/11. These exam tasks cover the first half of the class' curriculum, i.e. direct current circuits, electrostatics and magnetism. We can view exam tasks as objects with their attributes, and that makes them suitable for formal concept analysis. In this case the formal context consists of 10 exam tasks - objects, which are described with 20 different attributes. As stated before, we denote that an object has a certain attribute by placing an *"X"* mark in the

formal context matrix on the intersection of object's row and attribute's column. This formal context for the FCA method is shown in the Table 1.

Table 1. Formal context of a Fundamentals of E.E. midterm exam

	A1: el. field	**A2:** el. potential	**A3:** energy	**A4:** el. charge	**A5:** el. conductance	**A6:** el. resistance	**A7:** direct current	**A8:** capacity	**A9:** capacitor networks	**A10:** DC power	**A11:** DC networks	**A12:** current source	**A13:** voltage source	**A14:** real source	**A15:** mag. force	**A16:** Ohm' law	**A17:** Kirchhoff's laws	**A18:** DC voltage	**A19:** ideal source	**A20:** mag. field
T1	X	X	X	X																
T2					X	X														
T3		X				X	X							X		X		X		
T4		X	X			X	X			X	X	X				X	X	X	X	
T5		X		X			X	X	X				X					X	X	
T6		X				X	X			X	X		X			X	X	X	X	
T7		X				X	X				X		X			X	X	X	X	
T8		X				X	X				X	X				X	X	X	X	
T9		X				X	X				X	X				X	X	X	X	
T10				X			X								X					X

Using various tools (*Lattice Miner 1.4* and *Concept Explorer 1.3 – Java* applications under *Windows*; *Concepts 0.3* and *Graphplace* – programs under *Linux*) we will construct a concept lattice from the formal context in the Table 1.

We must stress that this kind of analysis could be very useful in e-learning systems for automated generation of homework assignments and exams. When the instructor sets the exam requirements (e.g. number of exam variations, number of questions, maximum number of different concepts exam can cover, which lessons exam covers), and the system would generate any number of requested exams from the questions/tasks database. FCA method can facilitate this process, because it would be very hard and time consuming to infer all the concepts and their superconcept–subconcept relationships only by using complex SQL queries on the relational database. But it also must be noted that, as a prerequisite for using FCA method, all the questions/tasks in the database must be semantically described with a list of their attributes. Manual assignment of attributes in a large questions' database is evidently very hard and inefficient, so it is important to find feasible ways of automating this job. In the field of intelligent tutoring systems some important advances have already been made towards an automated test generation from ontology–based knowledge [18,20].

5. Results

All the tools we used have conducted the FCA method similarly, and all have given, as expected, isomorphic concept lattices and the same number of identified concepts – 24. In the Fig. 2 the concept lattice generated by the program *Concept Explorer* is shown. For a clearer overview we have chosen the layout with the condensed information about every formal concept, i.e. the attributes are explicitly shown only in the first concept (implicitly they are also in all subsequent sub-concepts), and for each concept only a number and the percentage of their objects (exam tasks) instead of extensive object lists.

We can see from the concept lattice graphical layout, just like from the Table 1, that a vast majority of tasks requires students to know various different concepts. The only exception is the task **T2**. For its solution students only need to know two different concepts (electrical resistance and conductivity, which are related through a single

simple formula). In this case most of the exam tasks covered the direct current circuit theory, described with more attributes, and fewer (3/10) covered the theory of electrostatics and magnetism, described with fewer attributes. That made the resulting concept lattice asymmetric, with a dense left branch with direct circuit concepts and sparse right branch with electrostatics and magnetism concepts. It is clear that by increasing object and attribute sets we would find more new concepts (e.g. by defining new, more detailed attributes such as simple/complex direct current circuit or homogenous/heterogenous electric field).

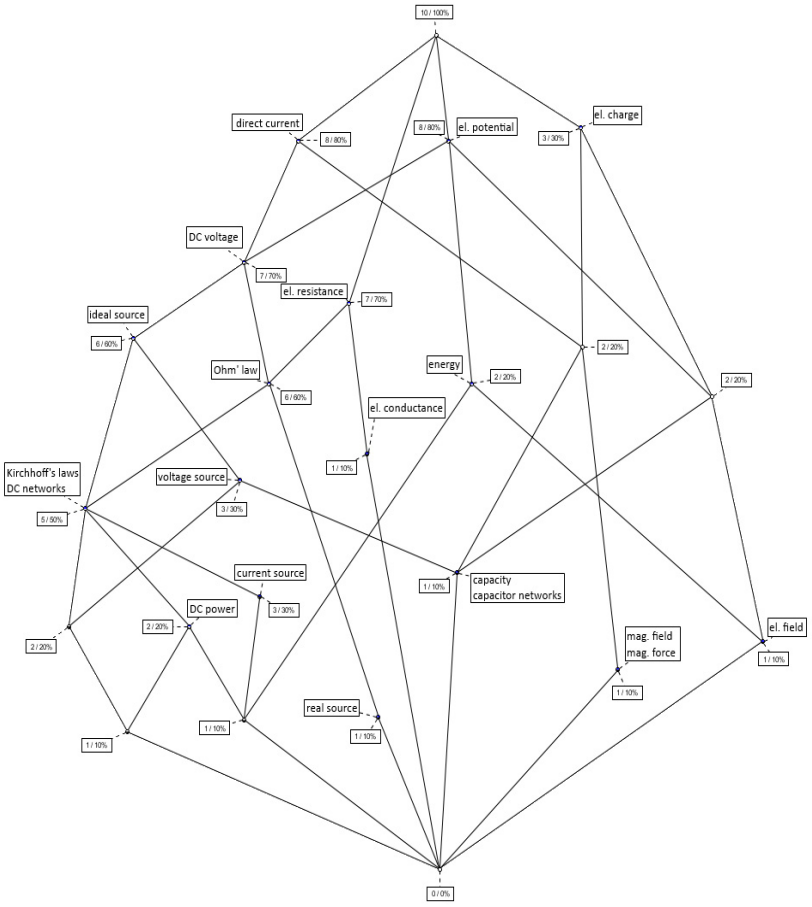

Fig. 2. concept lattice for the exam questions example

Some tools (*Lattice Miner 1.4*, *Concept Explorer 1.3*) generate not only the concept lattice, but also the list of found dependencies between the attributes, i.e. attribute implications or association rules. With the minimal support of just 10% (in this case only one object) and minimal confidence of 100% (only the clear implications between the attributes in the formal context) we found 42 association rules. Table 2 shows a list of 11 association rules with the greatest support (a minimal support of at least 50 %), e.g. rule 3 {*Ohm's law*}⇒{*DC voltage*, *el. potential*, *el. resistance*, *direct current*} (has a support of exactly 60% – it is true for 6 different tasks). Also, theory **T** contains 29 implications which represent the Guigues-Duquenne (irreducible) basis of the given domain.

Table 2. List of association rules (attribute implications) with a minimal support of 50%

	Attribute implications	Support
1.	**{A18}=>{A2, A7}**	70.0%
2.	**{A2, A7}=>{A18}**	70.0%
3.	**{A16}=>{A18, A2, A6, A7}**	60.0%
4.	**{A18, A6}=>{A16, A2, A7}**	60.0%
5.	**{A2, A6}=>{A16, A18, A7}**	60.0%
6.	**{A6, A7}=>{A16, A18, A2}**	60.0%
7.	**{A19}=>{A18, A2, A7}**	60.0%
8.	**{A11}=>{A16, A17, A18, A19, A2, A6, A7}**	50.0%
9.	**{A17}=>{A11, A16, A18, A19, A2, A6, A7}**	50.0%
10.	**{A16, A19}=>{A11, A17, A18, A2, A6, A7}**	50.0%
11.	**{A19, A6}=>{A11, A16, A17, A18, A2, A7}**	50.0%

We can also use concept lattice when we analyze the students' results on this exam. Overall exam score was 49%, or 40% when we take into account the negative points for wrong answers. All the tasks had a ideal completion rate of 40% – 60%, except for the tasks **T6** and **T4**. Task **T6** was a typical exam task and it had the best completion rate (64%), even though it covered 12 formal concepts. Task **T4** was unorthodox, it covered also 12 formal concepts, but it required more understanding and carefulness. That made **T4** the hardest task, with a completion rate of only 13%. Nevertheless, it can be argued that the tasks which cover more formal concepts will be more demanding for the students. Also, we should take other factors into considerations when analyzing exam results, e.g. students' previous results, exam environment [19].

As we discussed previously, the concept lattice is an acyclic directed graph (on the Fig. 2 all the edges are directed top to bottom, from the superconcepts to the subconcepts) so it is possible to apply a topological sorting algorithm (e.g. *Linux* program *tsort*) to the concept lattice in the Fig. 2. As a result, we got a list of all 24 found concepts, sorted from the superconcepts to the subconcepts, with respect to all relations between the concepts. These sorted concept lists can be used for choosing the best order of questions for exams. Also, it could be used in the e-learning systems, e.g. if a student fails to answer a question from the given sequence the system would redirect him to additional questions from that lesson and related previous lessons. And only when the student answers correctly to those questions it would be possible to proceed with the learning process.

In this example, using the program *tsort* we got two, almost identical, optimal orders of exam tasks. First optimal order of tasks is: **T2 – T1 – T10 – T3 – T5 – T7 – T8 – T9 – T6 – T4**, and the second is: **T2 – T1 – T10 – T3 – T5 – T7 – T9 – T8 – T6 – T4**. It is evident that the calculated optimal order of tasks is quite different from their actual order on the midterm exam (as shown in Table 1: **T1 – T2 – T3 – T4 – T5 – T6 – T7 – T8 – T9 – T10**). As expected, tasks with the greater extent are positioned near the end of the list, and the tasks with the smaller extent are positioned near the start of the list. It is worth nothing that the last task (**T4**) had the worst completion rate – only 13%, arguably because of many concepts it covers. Also, the first task (**T2**) had one of the best completion rates – 58%, because it was covered by only 3 formal concepts, which makes it one of the easiest tasks.

Thus, it is evident that the FCA method in this example resulted with a clear overview of the exam questions classification, which can be further used as a first step in the engineering of a complete ontology of this presented subdomain or even the whole domain of Fundamentals of electrical engineering theory.

6. Conclusion and future work

In this paper we have presented key ideas and described the main features of the FCA method. Finally, we have given a brief example showing the application of this method to the domain of computer aided learning, and we discussed its main goal, i.e. comprehensive categorization and classification of the domain knowledge. We have

indicated our principal paths for utilizing the FCA method – to facilitate the exam creation and to integrate it into e-learning systems.

The main advantages of the FCA method are the simplicity of preparing large input data sets for analysis, and that the analysis can result with the clear visual overview of the input domain and with a list of important attribute implications. Furthermore, the ongoing research work in this field is focused on the automatic creation of the input formal context from the text and the possibility of automatic ontology engineering. Also, there are proposals for linking the FCA method and context graphs and conceptual graphs, and also, the new temporal concept analysis is being developed by introducing the temporal attributes to the formal concept [21,6,7,8].

The goal of our future research work is to find concrete evidence which will prove the validity of the idea of using this method in the field of computer aided learning. We will try to find ways for constructing ontologies by combining the FCA method and description logic. Also, our intention is to use FCA method for finding optimal learning paths and for the quality control of the exam materials so they cover as many concepts form the given domain as possible.

References

[1] R. Wille, Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival editor: Ordered Sets, Dordrecht-Boston: Reidel; 1982, pp. 445–470.
[2] B. Ganter, G. Stumme, R. Wille editors. Formal Concept Analysis - Foundations and Applications. Berlin Heidelberg: Springer-Verlag; 2005.
[3] K.E. Wolff. A first course in Formal Concept Analysis. In: F. Faulbaum editor, StatSoft '93, Gustav Fischer Verlag, pp. 429–438.
[4] B. Ganter, R. Wille. Formal Concept Analysis- Mathematical Foundations. Berlin: Springer-Verlag; 1999.
[5] R. Belohlavek. Introduction to Formal Concept Analysis. belohlavek.inf.upol.cz/vyuka/IntroFCA.pdf, Olomuc; 2008.
[6] B. Ganter, R. Godin editors. Formal Concept Analysis, Third International Conference, ICFCA 2005. Springer, 2005.
[7] S.O. Kuznetsov, S. Schmidt editors. Formal Concept Analysis, Fifth International Conference, ICFCA 2007. Springer, 2007.
[8] S. Ferré, S. Rudolph editors. Formal Concept Analysis, Seventh International Conference, ICFCA 2009. Springer, 2009.
[9] S.O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. J. Univers. Comput. Sci. 2004;10(8):927–933.
[10] B. Ganter. Algorithmen zur Formalen Begriffsanalyse. In: B. Ganter, R. Wille, K. E. Wolff editors. Beiträge zur Begriffsanalyse, B. I. Wissenschaftsverlag; 1987, pp. 241–254.
[11] V. Duquenne, J.-L. Guigues. Famille minimale d'implications informatives resultant d'un tableau de donnes binaires. Math. et Sci. Hum. 1986;24(95):5–18.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. Section 22.4: Topological sort. In: Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill; 2001, pp. 549–552.
[13] W.W. Armstrong. Dependency structures of data base relationships. In: J.L. Rosenfeld, H. Freeman editors. Information Processing 74: Proceedings of IFIP Congress 74, North Holland; 1974, pp. 580–583.
[14] S.S. Skiena. Section 15.2: Topological Sorting. In: The Algorithm Design Manual (2nd ed.), London: Springer-Verlag; 2008, pp. 481–483.
[15] J.A. Bondy, U.S.R. Murty. Graph Theory. New York: Springer; 2008.
[16] D. Žubrnić. Diskretna matematika. Zagreb: Element; 2001.
[17] T.S. Blyth. Lattices and Ordered Algebraic Structures. London: Springer-Verlag; 2005.
[18] B. Žitko, S. Stankov, M. Rosić, A. Grubišić. Dynamic test generation over ontology-based knowledge representation in authoring shell. Expert Systems with Applications 2009;36(4):8185-8196.
[19] M. Vranić, D. Pintar, Z. Skočir. Data Mining and Statistical Analyses for High Education Improvement. In: D. Čišić, Ž. Hutinski et al. editors. Proceedings of 31st international convention on information and communication technology, electronics and microelectronics - MIPRO 2008, Zagreb, 2008, pp. 164-169.
[20] T. Alsubait, B. Parsia, U. Sattler. Next generation of e-assessment: automatic generation of questions. Int. J. of Technology Enhanced Learning 2012;4(3-4):156-171.
[21] M. Clark, Y. Kim, U. Kruschwitz, D. Song et al. Automatically structuring domain knowledge from text: An overview of current research. Inf. Process. Manage. 2012;48:552-568.