Pattern Structures for Analyzing Complex Data

Sergei O. Kuznetsov

Department of Applied Mathematics and Information Science,
State University Higher School of Economics,
Myasnitskaya 20, 101000 Moscow, Russia,
skuznetsov@hse.ru

Abstract. For data given by binary object-attribute datatables Formal Concept Analysis (FCA) provides with a means for both convenient computing hierarchies of object classes and dependencies between sets of attributes used for describing objects. In case of data more complex than binary to apply FCA techniques, one needs *scaling* (binarizing) data. Pattern structures propose a direct way of processing complex data such as strings, graphs, numerical intervals and other. As compared to scaling (binarization), this way is more efficient from the computational point of view and proposes much better vizualization of results. General definition of pattern structures and learning by means of them is given. Two particular cases, namely that of graph pattern structures and interval pattern structures are considered. Applications of these pattern structures in bioinformatics are discussed.

1 Introduction

Many problems of constructing domain taxonomies and ontologies, as well as finding dependencies in data, can be solved with the use of the models based on closure operators and respective lattices of closed sets within Formal Concept Analysis (FCA) [21,9]. The main definitions of FCA start from a binary relation, coming from applications as a binary object-attribute table. These tables (called *contexts* in FCA) give rise to lattices whose diagrams give nice visualizations of classes of objects of a domain. At the same time, the edges of these diagrams give essential knowledge about objects, by giving the probabilities of cooccurrence of attributes describing objects [17,18,19], this type of knowledge being known under the name of *association rules* in data mining.

However in many real-world applications researchers deal with complex and heteregeneous data different from binary datatables in involving numbers, strings, graphs, intervals, logical formulas, etc. for making descriptions of objects from an application domain. To apply FCA tools to data of these types, one needs binarizing initial data or, in FCA terms, applying *conceptual scaling*. Many types of scaling exist (see [9]), but do not always suggest the most efficient implementation right away, and there are situations where one would choose original or other data representation forms rather than scaled data [7]. Although scaling allows one to apply FCA tools, it may drastically increase the complexity of representation and worsen the visualization of results.

Instead of scaling one may work directly with initial data descriptions defining so-called *similarity* operators, which induce semilattice on data descriptions. In recent decades several attempts were done in defining such semilattices on sets of graphs [12,16,13], numerical intervals [12,10], logical formulas [2,3], etc. In [7] a general approach called pattern structures was proposed, which allows one to extend standard FCA approaches to arbitrary partially ordered data descriptions. In this paper we consider pattern structures for several data types and applications, showing their advantages and application potential.

The rest of the paper is organized as follows: In Section 2 we recall basic definitions of FCA, as well as related machine learning and rule mining models. In Section 3 we present pattern structures and respective generalization of machine learning and rule mining models. In Sections 4 and 5 we consider particular pattern structures on sets of graphs and vectors of intervals and discuss their applications in bioinformatics. In Section 6 we discuss computational issues of pattern structures.

2 Concept Lattices and Concept-Based Learning

2.1 Main definitions

First we introduce standard FCA definitions from [9]. Let G and M be arbitrary sets and $I \subseteq G \times M$ be an arbitrary binary relation between G and M. The triple (G, M, I) is called a *(formal) context.* Each $g \in G$ is interpreted as an object, each $m \in M$ is interpreted as an attribute. The fact $(g, m) \in I$ is interpreted as "g has attribute m". The two following *derivation operators* $(\cdot)'$

$$A' = \{m \in M \mid \forall g \in A : gIm\} \qquad \text{for } A \subseteq G,$$
$$B' = \{g \in G \mid \forall m \in B : gIm\} \qquad \text{for } B \subseteq M$$

define a *Galois connection* between the powersets of G and M. For $A \subseteq G$, $B \subseteq M$, a pair (A, B) such that $A' = B$ and $B' = A$, is called a *(formal) concept.* Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ $(\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I). For a concept (A, B) the set A is called the *extent* and the set B the *intent* of the concept.

The notion of dependency in data is captured in FCA by means of implications and partial implications (association rules). For $A, B \subseteq M$ the *implication* $A \to B$ holds if $A' \subseteq B'$ and the *association rule* (called *partial implication* in [17]) $A \longrightarrow_{c,s} B$ with *confidence* c and *support* s holds if $s \geq \frac{|A' \cap B'|}{|G|}$ and $c \geq \frac{|A' \cap B'|}{|A'|}$.

The language of FCA, as we showed in [6], is well suited for describing a model of learning JSM-hypotheses from [4,5]. In addition to the structural attributes of M, consider a *target attribute* $\omega \notin M$. This partitions the set G of all objects into three subsets: The set G_+ of those objects that are known to have the property ω (these are the *positive examples*), the set G_- of those objects of which it is known that they do not have ω (the *negative examples*) and the set G_τ of *undetermined examples*, i.e., of those objects, of which it is unknown if they have property ω or not. This gives three subcontexts of $\mathbb{K} = (G, M, I)$, the first two staying for the training sample:

$$\mathbb{K}_+ := (G_+, M, I_+), \quad \mathbb{K}_- := (G_-, M, I_-), \quad \text{and } \mathbb{K}_\tau := (G_\tau, M, I_\tau),$$

where for $\varepsilon \in \{+, -, \tau\}$ we have $I_\epsilon := I \cap (G_\varepsilon \times M)$ and the corresponding derivation operators are denoted by $(\cdot)^+$, $(\cdot)^-$, $(\cdot)^\tau$, respectively.

Intents, as defined above, are attribute sets shared by some of the observed objects. In order to form hypotheses about structural causes of the target attribute ω, we are interested in sets of structural attributes that are common to some positive, but to no negative examples. Thus, a *positive hypothesis* h for ω (called "counter-example forbidding hypotheses" in the JSM-method [4,5]) is an intent of $\mathbb{K}_+$ such that $h^+ \neq \emptyset$ and $h \not\subseteq g^- := \{m \in M \mid (g, m) \in I_-\}$ for any negative example $g \in G_-$. *Negative hypotheses* are defined similarly. Various classification schemes using hypotheses are possible, as an example consider the following simple scheme from [5]: If the intent

$$g^\tau := \{m \in M \mid (g, m) \in I_\tau\}$$

of an object $g \in G_\tau$ contains a positive, but no negative hypothesis, then g^τ is *classified positively*. Negative classifications are defined similarly. If g^τ contains hypotheses of both kinds, or if g^τ contains no hypothesis at all, then the classification is contradictory or undetermined, respectively. In this case one can apply standard probabilistic techniques known in machine learning and data mining (majority vote, Bayesian approach, etc.). Notwithstanding its simplicity, the model of learning and classification with concept-based hypotheses proved to be efficient in numerous studies in bioinformatics [1,8,15].

A well-known application of concept lattices in data mining use the fact that the edges of the lattice diagram make a basis of association rules for the context [17,18,19]. In fact, each edge of a concept lattice diagram, connecting a higher concept (A', A) and a lower concept (B, B'), corresponds to a set of association rules of the form $(Y) \longrightarrow_{c,s} B$ (where Y is minimal in the set $\{X \subseteq A \mid X'' = A\}$) and all other association rules may be obtained from rules of these type by some inference [11].

2.2 Many-valued contexts and their interordinal scaling

Consider an object-attribute table whose entries are not binary. It can be given by a quadruple $\mathbb{K}_1 = (G, S, W, I_1)$, where G, S, W are sets and I_1 is a ternary relation $I_1 \subseteq G \times S \times W$. In FCA terms $\mathbb{K}_1 = (G, S, W, I_1)$ is called a *many-valued context.*

Consider an example of analyzing *gene expression data* (GED) given by tables of values. The names of rows correspond to genes. The names of the columns of the table correspond to *situations* where genes are tested. A table entry is called an *expression value.* A row in the table is called *expression profile* associated to a gene. In terms of many-valued contexts, the set of genes makes the set of objects G, the set of situations makes the set of many-valued attributes S, the set of expression values makes the set $W \subset \mathbb{R}$ and $I_1 \subseteq G \times S \times W$. Then $\mathbb{K}_1 = (G, S, W, I_1)$ is a many-valued context representing a GED. The fact $(g, s, w) \in I_1$ or simply $g(s) = w$ means that gene g has an expression value w for situation s. The objective is to extract formal concepts (A, B) from $\mathbb{K}_1$, where $A \subseteq G$ is a subset of genes sharing "similar values" of W, i.e. lying in a same interval. To this end, we use an appropriate binarization (scaling) technique to build a formal context $\mathbb{K}_2 = (G, S_2, I_2)$, called *derived context* of $\mathbb{K}_1$.

A scale is a formal context (cross-table) taking original attributes of $\mathbb{K}_1$ with the derived ones of $\mathbb{K}_2$. As attributes do not take necessarily same values, each of them is scaled separately. Let $W_s \subseteq W$ be the set of all values of the attribute s. The following *interordinal scale* (see pp. 42 in [9]) can be used to represent all possible intervals of attribute values:

$$\mathbb{I}_{W_s} = (W_s, W_s, \leq)|(W_s, W_s, \geq).$$

The operation of *apposition of two contexts* with identical sets of objects, denoted by $|$, returns the context with the same set of objects W_s and the set of attributes corresponding to the disjoint union of attribute sets of the original contexts. In our case this operation is applied to two contexts $(W_s, W_s, \leq)$ and $(W_s, W_s, \geq)$), the table below gives an example for $W_s = \{4, 5, 6\}$.

	$s_1 \leq 4$	$s_1 \leq 5$	$s_1 \leq 6$	$s_1 \geq 4$	$s_1 \geq 5$	$s_1 \geq 6$
4	×	×	×	×		
5		×	×	×	×	
6			×	×	×	×

The intents given by interordinal scaling are value intervals.

3 Pattern Structures

3.1 Main definitions and results

Let G be a set (interpreted as a set of objects), let $(D, \sqcap)$ be a meet-semi-lattice (of potential object descriptions) and let $\delta : G \longrightarrow D$ be a mapping. Then $(G, \underline{D}, \delta)$, where $\underline{D} = (D, \sqcap)$, is called a *pattern structure*, provided that the set

$$\delta(G) := \{\delta(g) \mid g \in G\}$$

generates a complete subsemilattice $(D_\delta, \sqcap)$ of $(D, \sqcap)$, i.e., every subset X of $\delta(G)$ has an infimum $\sqcap X$ in $(D, \sqcap)$ and D_δ is the set of these infima.

Elements of D are called *patterns* and are naturally ordered by subsumption relation $\sqsubseteq$: given $c, d \in D$ one has $c \sqsubseteq d \iff c \sqcap d = c$. A pattern structure $(G, \underline{D}, \delta)$ gives rise to the following derivation operators $(\cdot)^\diamond$:

$$A^\diamond = \bigsqcap_{g \in A} \delta(g) \qquad \text{for } A \subseteq G,$$

$$d^\diamond = \{g \in G \mid d \sqsubseteq \delta(g)\} \qquad \text{for } d \in (D, \sqcap).$$

These operators form a Galois connection between the powerset of G and $(D, \sqsubseteq)$. $\sqcap$ is also called a similarity operator. The pairs (A, d) satisfying

$$A \subseteq G, \quad d \in D, \quad A^\diamond = d, \quad \text{and} \quad A = d^\diamond$$

are called the *pattern concepts* of $(G, \underline{D}, \delta)$, with extent A and *pattern intent* d. For $a, b \in D$ the *pattern implication* $a \rightarrow b$ holds if $a^\diamond \subseteq b^\diamond$, and the *pattern association rule* $a \longrightarrow_{c,s} b$ with *confidence* c and *support* s holds if $s \geq \frac{|a^\diamond \sqcap b^\diamond|}{|G|}$ and $c \geq \frac{|a^\diamond \sqcap b^\diamond|}{|a^\diamond|}$. Like in case of association rules, pattern association rules may be inferred from a base that corresponds to the set of edges of the diagram of the pattern concept lattice.

Operator $(\cdot)^{\diamond\diamond}$ is an algebraical closure operator [9] on patterns, since it is

idempotent: $d^{\diamond\diamond\diamond\diamond} = d^{\diamond\diamond}$,
extensive: $d \sqsubseteq d^{\diamond\diamond}$,
monotone: $d^{\diamond\diamond} \sqsubseteq c^{\diamond\diamond}$ for $d \sqsubseteq c$.

In [6] we showed that if $(D, \sqcap)$ is a complete meet-semi-lattice (where infimums are defined for arbitrary subsets of elements), in particular a finite semilattice, there is a subset $M \subseteq D$ with the following interesting property: The concepts of the formal context (G, M, I) where I is given as $gIm\colon \Leftrightarrow m \sqsubseteq \delta(g)$, called a *representation context* for $(G, \underline{D}, \delta)$, are in one-to-one correspondence with the pattern concepts of $(G, \underline{D}, \delta)$. The corresponding concepts have the same first components (called *extents*). These extents form a complete lattice, which is isomorphic to the concept lattice of (G, M, I). This result is proved by a standard application of the basic theorem of FCA (which allows one to represent every lattice as a concept lattice) [21,9] and shows the way of binarizing complex data representation given by a pattern structure. The cost of this binarization may be a large amount attributes of the representation context and hence, the space needed for storing this context.

3.2 Learning with pattern structures

The concept learning model described in the previous section for standard object-attribute representation (i.e., formal contexts) is naturally extended to pattern structures. Suppose we have a set of positive examples E_+ and a set of negative examples E_- w.r.t. a target attribute.

A pattern $h \in D$ is a *positive hypothesis* iff

$$h^\diamond \cap E_- = \emptyset \text{ and } \exists A \subseteq E_+ : A^\diamond = h.$$

Again, a positive hypothesis is a similarity (or least general generalization of descriptions) of positive examples, which is not contained in (does not cover) any negative example. A *negative hypothesis* is defined analogously, by interchanging $+$ and $-$.

The meet-preserving property of projections implies that a hypothesis H_p in data under projection ψ corresponds to a hypothesis H in the initial representation for which the image under projection is equal to H_p, i.e., $\psi(H) = H_p$.

Hypotheses are used for classification of undetermined examples along the lines of [5]. The corresponding definitions are similar to those from Section 2, one just needs to replace $\subseteq$ with $\sqsubseteq$.

3.3 Projections and learning in projections

For some pattern structures (e.g., for the pattern structures on sets of graphs with labeled vertices) even computing subsumption of patterns may be NP-hard. Hence, for practical situations one needs approximation tools, which would replace the patterns with simpler ones, even if that results in some loss of information. To this end we use a mapping $\psi\colon D \to D$ that replaces each pattern $d \in D$ by $\psi(d)$ such that the pattern structure $(G, \underline{D}, \delta)$ is replaced by $(G, \underline{D}, \psi \circ \delta)$. To distinguish two pattern structures, which we consider simultaneously, we use the symbol $^\diamond$ only for $(G, \underline{D}, \delta)$, not for $(G, \underline{D}, \psi \circ \delta)$. Under some natural algebraic requirements (that hold for all natural projections in particular pattern structures we studied in applications) the meet operation $\sqcap$ is preserved:

$$\psi(X \sqcap Y) = \psi(X) \sqcap \psi(Y).$$

This property of projection allows one to relate hypotheses in the original representation with those approximated by a projection.

This helped us to describe [6] how the lattice of pattern concepts changes when we replace $(G, \underline{D}, \delta)$ by its approximation $(G, \underline{D}, \psi \circ \delta)$. First, we note that $\psi(d) \sqsubseteq \delta(g) \Leftrightarrow \psi(d) \sqsubseteq \psi \circ \delta(g)$. Moreover, for pattern structures $(G, \underline{D}, \delta_1)$ and $(G, \underline{D}, \delta_2)$ one has $\delta_2 = \psi \circ \delta_1$ for some projection ψ of $\underline{D}$ iff there is a representation context (G, M, I) of $(G, \underline{D}, \delta_1)$ and some $N \subseteq M$ such that $(G, N, I \cap (G \times N))$ is a representation context of $(G, \underline{D}, \delta_2)$. Thus, the basic theorem of FCA helps us not only to "binarize" the initial data representation, but to relate binarizations of different projections.

Pattern structures are naturally ordered by projections: $(G, \underline{D}, \delta_1) \geq (G, \underline{D}, \delta_2)$ if there is a projection ψ such that $\delta_2 = \psi \circ \delta_1$. In this case, representation $(G, \underline{D}, \delta_2)$ can be said to be rougher than $(G, \underline{D}, \delta_1)$ and the latter to be finer than the former. In comparable pattern structures implications are related as follows: If $\psi(a) \to \psi(b)$ and $\psi(b) = b$ then $a \to b$ for arbitrary $a, b \in D$. In particular, if $\psi(a)$ is a positive (negative) hypothesis in projected representation, then a is positive (negative) hypothesis in the original representation.

4 Pattern Structures on Closed Sets of Labeled Graphs

In [12,13] we proposed a semi-lattice on sets of graphs with labeled vertices and edges. This lattice is based on a natural domination relation between pairs of graphs with labeled vertices and edges. Consider an ordered set P of connected graphs[1] with vertex and edge labels from the set $\mathcal{L}$ partially ordered by $\preceq$. Each

[1] Omitting the condition of connectedness, one obtains a similar, but computationally much harder model.

labeled graph Γ from P is a quadruple of the form $((V, l), (E, b))$, where V is a set of vertices, E is a set of edges, $l \colon V \to \mathcal{L}$ is a function assigning labels to vertices, and $b \colon E \to \mathcal{L}$ is a function assigning labels to edges. In $(P, \leq)$ we do not distinguish isomorphic graphs.
For two graphs $\Gamma_1 := ((V_1, l_1), (E_1, b_1))$ and $\Gamma_2 := ((V_2, l_2), (E_2, b_2))$ from P we say that Γ_1 *dominates* Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a *subgraph* of Γ_1) if there exists an injection $\varphi \colon V_2 \to V_1$ such that it

– respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
– fits under labels: $l_2(v) \preceq l_1(\varphi(v))$, if $(v, w) \in E_2$ then $b_2(v, w) \preceq b_1(\varphi(v), \varphi(w))$.

Obviously, $(P, \leq)$ is a partially ordered set. Now a *similarity operation* $\sqcap$ on graph sets can be defined as follows: For two graphs X and Y from P

$$\{X\} \sqcap \{Y\} := \{Z \mid Z \leq X, Y, \ \forall Z_* \leq X, Y \ Z_* \not\geq Z\},$$

i.e., $\{X\} \sqcap \{Y\}$ is the set of all maximal common subgraphs of graphs X and Y. Similarity of non-singleton sets of graphs $\{X_1, \ldots, X_k\}$ and $\{Y_1, \ldots, Y_m\}$ is defined as

$$\{X_1, \ldots, X_k\} \sqcap \{Y_1, \ldots, Y_m\} := \mathrm{MAX}_{\leq}(\cup_{i,j}(\{X_i\} \sqcap \{Y_j\})),$$

where $\mathrm{MAX}_{\leq}(X)$ returns maximal (w.r.t. $\leq$) elements of X.

The similarity operation $\sqcap$ on graph sets is commutative: $X \sqcap Y = Y \sqcap X$ and associative: $(X \sqcap Y) \sqcap Z = X \sqcap (Y \sqcap Z)$. A set X of labeled graphs from P for which $\sqcap$ is idempotent, i.e., $X \sqcap X = X$ holds, is called a *graph pattern*. For patterns we have $\mathrm{MAX}_{\leq}(X) = X$. For example, for each graph $g \in P$ the set $\{g\}$ is a pattern. On the contrary, for $\Gamma_1, \Gamma_2 \in P$ such that $\Gamma_1 \leq \Gamma_2$ the set $\{\Gamma_1, \Gamma_2\}$ is not a pattern. Denote by D the set of all patterns, then $(D, \sqcap)$ is a semi-lattice with infimum (meet) operator $\sqcap$. The natural subsumption order on patterns is given by $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$.

Let E be a set of object names, and let $\delta : E \to D$ be a mapping, taking each object name to $\{g\}$ for some labeled graph $g \in P$ (thus, g is "graph description" of object e). The triple $(E, (D, \sqcap), \delta)$ is a particular case of a pattern structure.

A set of graphs X is called *closed* if $X^{\diamond\diamond} = X$. This definition is related to the notion of a closed graph in data mining and graph mining, which is important for computing association rules between graphs. Closed graphs are defined in [20] in terms of "counting inference" as follows.

Given a graph dataset E, support of a graph g or *support*(g) is a set (or number) of graphs in E that have subgraphs isomorphic to g. A graph g is called *closed* if no supergraph f of g (i.e., a graph such that g is isomorphic to its subgraph) has the same support.

In terms of pattern structures, E is a set of objects, each object $e \in E$ having a graph description $\delta(e)$, $\mathrm{support}(g) = \{e \in E \mid \delta(g) \leq e\}$. Note that the definitions distinguish between a closed graph g and the closed set $\{g\}$ consisting of one graph g. Closed sets of graphs form a *meet semi-lattice* w.r.t. $\sqcap$. Closed graphs do not have this property, since in general, there are closed graphs with no infimums. However, closed graphs and closed sets of graphs are intimately related, as shown in the following

Proposition 1 *Let a dataset described by a pattern structure* $(E, (D, \sqcap), \delta)$ *be given. Then the following two properties hold:*

1. *For a closed graph* g *there is a closed set of graphs* G *such that* $g \in G$.
2. *For a closed set of graphs* G *and an arbitrary* $g \in G$, *graph* g *is closed.*

Proof. 1. Consider the closed set of graphs $G = \{g\}^{\diamond\diamond}$. Since G consists of all maximal common subgraphs of graphs that have g as a subgraph, G contains as an element either g or a supergraph f of g. In the first case, property 1 holds. In the second case, we have that each graph in G that has g as a subgraph also has f as a subgraph, so f has the same support as g, which contradicts with the fact that g is closed. Thus, $G = \{g\}^{\diamond\diamond}$ is a closed set of graphs satisfying property 1.
2. Consider a closed set of graphs G and $g \in G$. If g is not a closed graph, then there is a supergraph f of it with the same support as g has and hence, with the same support as G has. Since G is the set of all maximal common subgraphs of the graphs describing examples from the set $G^{\diamond}$ (i.e, its support), $f \in G$ should hold. This contradicts the fact that $g \in G$, since a closed set of graphs cannot contain as elements a graph and a supergraph of it (otherwise, its closure does not coincide with itself). $\square$

Therefore, one can use algorithms for computing closed sets of graphs, e.g., the algorithm described in [13], to compute closed graphs. With this algorithm one can also compute all *frequent* closed sets of graphs, i.e., closed sets of graphs with support above a fixed threshold (by introducing a slightly different backtrack condition).

The learning model based on graph pattern structures along the lines of the previous section was successfully used in series of applications in bioinformatics, namely in problems where chemical substructures causing particular biological activities (like toxicity) were investigated [8,15]. In many cases the proposed graph representation resulted in better predictive accuracy as compared to that obtained with standard attribute-type languages used for the analysis of biological activity of chemicals.

5 Pattern Structures on Intervals

5.1 Main definitions

To define a semilattice operation $\sqcap$ for intervals that would be analogous to the set-theoretic intersection or meet operator on sets of graphs, one should realize that "similarity" between two real numbers (between two intervals) may be expressed in the fact that they lie within some (larger) interval, this interval being the smallest interval containing both two.

Thus, for two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$, we define their meet as

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)].$$

This operator is obviously idempotent, commutative and associative, thus defining a pattern structure on intervals. The counterintuitive observation that the meet operator takes two intervals to a larger one (in contrast to set intersection and meet on graph sets which take sets to smaller ones) fails after realizing that a larger interval, like in case of smaller sets and smaller sets of graphs, correspond to a larger set of objects, whose descriptions fall in the interval.

The natural order relation (subsumption) on intervals is given as follows:

$$\begin{aligned} &[a_1, b_1] \sqsubseteq [a_2, b_2] \\ \Longleftrightarrow\ &[a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \\ \Longleftrightarrow\ &[min(a_1, a_2), max(b_1, b_2)] = [a_1, b_1] \\ \Longleftrightarrow\ &a_1 \leq a_2 \quad and \quad b_1 \geq b_2. \end{aligned}$$

Again, contrary to usual intuition, smaller intervals subsume larger intervals that contain the former. A next step would be considering vectors of intervals. An *interval p-vector* is a p-dimensional vector of intervals. The meet $\sqcap$ for interval vectors is defined by component-wise interval meets. Interval p-vector patterns are p-dimensional rectangular parallelepipeds in Euclidean space. Another step further would be made by allowing any type of patterns for each component. The general meet operator on a vector like that is defined by component-wise meet operators.

5.2 Interval patterns and interordinal scaling

For a many-valued context (G, M, W, I) with $W \subset \mathbb{R}$ consider the respective pattern structure $(G, (D, \sqcap), \delta)$ on interval vectors, the interordinal scaling $\mathbb{I}_{W_s} = (W_s, W_s, \leq) \mid (W_s, W_s, \geq)$ from the previous Section, and the context K_I resulting from applying interordinal scaling $\mathbb{I}_{W_s}$ to (G, M, W, I). Consider usual derivation operators $(\cdot)'$ in context K_I. Then the following proposition establishes an isomorphism between the concept lattice of K_I and the pattern concept lattice of $(G, (D, \sqcap), \delta)$.

Proposition 2 *Let $A \subseteq G$, then the following statements 1 and 2 are equivalent:*

1. A is an extent of the pattern structure $(G, (D, \sqcap), \delta)$ and $A^\diamond = \langle[\underline{m}_i, \overline{m}_i]\rangle_{i \in [1,p]}$

2. A is a concept extent of the context K_I so that for all $i \in [1, p]$ $\underline{m}_i$ is the largest number n such that the attribute $s_i \geq n$ is in A' and $\overline{m}_i$ is the smallest number n such that the attribute $s_i \leq n$ is in A'.

Proof. $1 \rightarrow 2$ Let $A \subseteq G$ be a pattern extent. Given $\delta_i(g)$ the mapping that returns the i^{th} interval of the vector describing object g. Since $A^\diamond = \langle[\underline{m}_i, \overline{m}_i]\rangle_{i\in[1,p]}$, for every object $g \in A$ one has $\underline{m}_i \leq \delta_i(g) \leq \overline{m}_i$ and there are objects $g_1, g_2 \in A$ such that $\delta_i(g_1) = \underline{m}_i$, $\delta_i(g_1) = \overline{m}_i$. Hence, in context K_I one has

$$A' = \cup_{i\in[1,p]}\{s_i \geq n_{\text{min}}, \ldots, s_i \geq n_1, s_i \leq n_2, \ldots, s_i \leq n_{\text{max}}\}$$

where

$$n_{\text{min}} \prec \ldots \prec n_1 \leq n_2 \prec \ldots \prec n_{\text{max}}$$

and $n_1 = \underline{m}_i$, $n_2 = \overline{m}_i$. Hence, $\underline{m}_i$ is the largest number n such that the attribute $s_i \geq n$ is in A' and $\overline{m}_i$ is the smallest number n such that the attribute $s_i \leq n$ is in A'. Suppose that A is not an extent of K_I. Hence, $A \subset A''$ and there is $g \in A'' \setminus A$ and $g' \supseteq A'$. This means that for all i $\underline{m}_i \leq \delta_i(g) \leq \overline{m}_i$. Therefore, $g \in A^{\diamond\diamond}$ and $A \neq A^{\diamond\diamond}$, a contradiction. The proof $2 \rightarrow 1$ is similar. $\square$

The larger is a pattern concept, the more there are elements in its extent, and the more there are intervals in its intent. However, the main goal in applications like analysis of gene expression data is extracting homogeneous groups of objects (e.g., genes), i.e. groups of objects having similar expression values. Therefore, descriptions of homogeneous groups should be composed of intervals with "small" sizes where $size([a, b]) = b - a$. Consider parameter max_{size} that specifies the maximal admissible size of any interval composing an interval vector. In our gene expression data analysis [10] we restricted to pattern concepts with pattern intents $d = \langle[a_i, b_i]\rangle_{i\in[1,p]} \in (D, \sqcap)$ satisfying the constraint: $\exists i \in [1, p]$ $(b_i - a_i) \leq max_{size}$, for any $a, b \in \mathbb{R}$, or a stricter constraint like $\forall i \in [1, p]$ $(b_i - a_i) \leq max_{size}$, where max_{size} is a parameter. Since both constraints are monotone (if an intent does not satisfy it, than a subsumed intent does not satisfy it too), the subsets of patterns satisfying any of these constraints make an order filter (w.r.t. subsumption on intervals $\sqsubseteq$) of the lattice of pattern intents and can be computed by an ordinary FCA algorithm with a modified backtracking condition.

Interval pattern structures were successfully applied to gene expression data analysis [10], where classes of situations with similar gene expressions were generated.

5.3 Computing in pattern structures

Many algorithms for generating formal concepts from a formal context are known, see e.g. a performance comparative [14]. Experimental results of [14] highlight several best algorithms for dense and large contexts, which is the case of interordinal derived formal contexts. Worst-case upper bound time complexity of these algorithms computing the set of all concepts of the context (G, M, I) is $O(|G|^2 \cdot |M| \cdot |L|)$, where L is the set of generated concepts [14].

Several algorithms for computing concept lattices, like NextClosure and CbO, may be adapted to computing pattern lattices in bottom-up way (starting from intersecting individual object descriptions and proceeding by intersecting more and more object descriptions). The worst-case time complexity of computing all pattern concepts of a pattern structure $(G, \underline{D}, \delta)$ in the bottom-up way is $O((\alpha + \beta|G|)|G||L|)$, where α is time needed to perform $\sqcap$ operation and β is time needed to test $\sqsubseteq$ relation. In case of graphs, even β may be exponential wrt. the number of graph vertices, that is why approximations (like those given by projections) are often needed. In experiments with many chemical rows in [15] we used projections to graphs with about 10 vertices to be able to process datasets with hundreds of chemical substances.

The worst-case time complexity of computing the set of interval pattern structures is $O(|G|^2 \cdot |M| \cdot |L|)$. If a many-valued context (G, M, W, I) is given, the worst-case complexity of computing the set of all concepts of its interordinally scaling is $O(|G|^2 \cdot |W| \cdot |L|)$, which may be fairly large if the cardinality of the set of attribute values $|W|$ is much larger than that of the set of attributes $|M|$. The worst case $|W| = |G| \times |S|$ is attained when attribute values are different for each object-attribute pair. In [10] several algorithms for computing with interval patterns were compared. The experimental comparison shows that when the number of attribute values w.r.t. $|G| \times |S|$ is very low, computing concepts in representation contexts is more efficient. For large datasets with many different attribute values, it is more efficient to compute in pattern structures.

6 Conclusion

Pattern structures propose a universal means of analyzing hierarchies of classes and dependencies in case of data given by complex ordered descriptions. As compared to binarization techniques, computing with pattern structures often gives more efficiency and better vizualization. Pattern projections allows one to reduce representation dimension to attain even better computer efficiency. Future research on pattern structure will be concerned with new complex data types, interesting projections and new applications. The use of pattern structures for mining association rules in complex data will also be studied.

Acknowledgements

The author was partially supported by the project of the Russian Foundation for Basic Research, grant no. 08-07-92497-NTsNIL_a.

References

1. V.G. Blinova, D.A. Dobrynin, V.K. Finn, S.O. Kuznetsov, and E.S. Pankratova, Toxicology analysis by means of the JSM-method, *Bioinformatics*, 2003, **19**, pp. 1201–1207.
2. L. Chaudron and N. Maille, Generalized Formal Concept Analysis, in *Proc. 8th Int. Conf.on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 357-370.

3. S. Férré and O. Ridoux, A Logical Generalization of Formal Concept Analysis, in *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000.
4. V.K. Finn, On Machine-Oriented Formalization of Plausible Reasoning in the Style of F. Backon–J. S. Mill, *Semiotika Informatika*, **20** (1983) 35-101 [in Russian].
5. V.K. Finn, Plausible Reasoning in Systems of JSM Type, *Itogi Nauki i Tekhniki, Seriya Informatika*, **15**, 54-101, 1991 [in Russian].
6. B. Ganter and S. Kuznetsov, Formalizing Hypotheses with Concepts, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'00*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 342-356.
7. B. Ganter and S. Kuznetsov, Pattern Structures and Their Projections, *Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01*, G. Stumme and H. Delugach, Eds., Lecture Notes in Artificial Intelligence, **2120** 2001, pp. 129-142.
8. B. Ganter, P.A. Grigoriev, S.O. Kuznetsov, and M.V. Samokhin, Concept-based Data Mining with Scaled Labeled Graphs, *Proc. 12th Int. Conf. on Conceptual Structures, ICCS'04*, K.E. Wolff, H. D. Pfeiffer, H. S. Delugach, Eds., Lecture Notes in Artificial Intelligence, **3127** 2004 pp. 94-108.
9. B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.
10. Mehdi Kaytoue, Seabstien Duplessis, Sergei O. Kuznetsov, and Amedeo Napoli, Two FCA-Based Methods for Mining Gene Expression Data, *Proc. 7th Int. Conf. on Formal Concept Analysis (ICFCA2009)*, S.Ferre, S.Rudoplh, Eds., Lecture Notes in Artificial Intelligence, **5548** 2009, pp. 251-266.
11. M. Kryszkiewicz, Concise Representations of Association Rules, in *Pattern Detection and Discovery*, David J. Hand, Niall M. Adams, Richard J. Bolton (Eds.), Lecture Notes in Computer Science **2447**, 2002, pp.92-109.
12. S.O. Kuznetsov, JSM-method as a machine learning method, *Itogi Nauki i Tekhniki, ser. Informatika*, vol. 15, pp.17-50, 1991 [in Russian].
13. S.O. Kuznetsov, Learning of Simple Conceptual Graphs from Positive and Negative Examples. In: J. Zytkow, J. Rauch (eds.), Proc. *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD'99*, Lecture Notes in Artificial Intelligence, **1704**, pp. 384-392, 1999.
14. S.O. Kuznetsov, S.A. Obiedkov, Comparing performance of algorithms for generating concept lattices, *J. Exp. Theor. Artif. Intell.*, 2002, vol. 14, nos. 2-3, pp. 189-216.
15. S.O. Kuznetsov and M.V. Samokhin, Learning Closed Sets of Labeled Graphs for Chemical Applications, in *Proc. 15th Conference on Inductive Logic Programming (ILP2005)*, Lecture Notes in Artificial Intelligence, **3625**, 2005, pp.190-208.
16. M. Liquiere and J. Sallantin, Structural Machine Learning with Galois Lattice and Graphs, *Proc. Int. Conf. Machine Learning ICML'98*, 1998.
17. M. Luxenburger, Implications partielle dans un contexte, *Math. Sci. Hum.*, 1991.
18. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, Efficient Minining of Association Rules Based on Using Closed Itemset Lattices, *J. Inf. Systems*, **24**, 1999, pp. 25-46.
19. L. Lakhal, G. Stumme, in B. Ganter, G. Stumme, R. Wille, Eds., *Formal Concept Analysis: Foundations and Applications*, Lecture Notes in Artificial Intelligence, State-of-the Art Ser., 2005, vol. 3626, pp.
20. X.Yan, J. Han: CloseGraph: Mining closed frequent graph patterns, in L.Getoor, T.Senator, P.Domingos, C.Faloutsos, Eds., *Proc. 9th ACM SIGKDD Int.Conf. on Knowledge Discovery and Data Mining (KDD'03)*, ACM Press (2003) 286-295.
21. R. Wille, Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts, In: *Ordered Sets* (I. Rival, ed.), Reidel, Dordrecht–Boston, 445-470, 1982.