Sampling Representation Contexts with Attribute Exploration

Victor Codocedo, Jaume Baixeries, Mehdi Kaytoue, Amedeo Napoli

▶ **To cite this version:**

Victor Codocedo, Jaume Baixeries, Mehdi Kaytoue, Amedeo Napoli. Sampling Representation Contexts with Attribute Exploration. ICFCA 2019 - 15th International Conference on Formal Concept Analysis, Jun 2019, Frankfurt, Germany. pp.307-314, 10.1007/978-3-030-21462-3_20 . hal-02195498

HAL Id: hal-02195498
https://inria.hal.science/hal-02195498

Submitted on 10 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sampling Representation Contexts with Attribute Exploration

Victor Codocedo[1], Jaume Baixeries[3], Mehdi Kaytoue[2], and Amedeo Napoli[4]

[1] Departamento de Informática. Universidad Técnica Federico Santa María. Campus San Joaquín, Santiago de Chile.
[2] Université de Lyon. CNRS, INSA-Lyon, LIRIS. UMR5205, F-69621, France.
[3] Computer Science Department. Universitat Politècnica de Catalunya. 08032, Barcelona. Catalonia.
[4] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France

Corresponding author: victor.codocedo@inf.utfsm.cl

Abstract. We tackle the problem of constructing the representation context of a pattern structure. First, we present a naive algorithm that computes the representation context a pattern structure. Then, we add a sampling technique in order to reduce the size of the output. We show that these techniques reduce significantly the size of the representation context for interval pattern structures.

1 Introduction

Formal Concept Analysis (FCA) has dealt with non binary data mainly in two different ways: either by *scaling* [4] the original dataset into a formal context or through the use **pattern structures** [5,8]. The former is known to induce large formal contexts which are difficult to handle. The latter is well preferred for a wide variety of applications [6].

Any pattern structure can be represented by an equivalent formal context, which is consequently called a **representation context** (RC) [1,5]. An RC and a pattern structure contain the same set of extents. In this short paper we propose a general approach to build an RC for a given pattern structure using a sampling strategy based on attribute exploration. Our proposition aims at calculating a reduced RC as well as reducing the computational complexity of calculating pattern structures with high dimensional object representations.

2 Notation and Theoretical Background

In the following we introduce some definitions needed for the development of this article. The notations used are based on [4]. A formal context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ is a triple where $\mathtt{G}$ is a set of objects, $\mathtt{M}$ is a set of attributes and $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M}$ is an incidence relation with $(g, m) \in \mathtt{I}$ denoting "*object g has the attribute m*". The derivation operators are denoted as $' : \wp(\mathtt{G}) \rightarrow \wp(\mathtt{M})$ and $' : \wp(\mathtt{M}) \rightarrow \wp(\mathtt{G})$. A many-valued context $\mathcal{M} = (\mathtt{G}, \mathtt{N}, \mathtt{W}, \mathtt{J})$ is a data table where, in addition to $\mathtt{G}$ and $\mathtt{M}$, we define

a set of values $\mathtt{W}^m$ for each attribute $m \in \mathtt{M}$ (where $\mathtt{W} = \bigcup \mathtt{W}^m$ for all $m \in \mathtt{M}$) such that $m(g) = w$ denotes that *"the value of the attribute m for the object g is w"* (with $w \in \mathtt{W}^m$). Additionally, in this document we will consider each $\mathtt{W}^m$ as an ordered set where $\mathtt{W}^m_i$ denotes the i-th element in the set.

The pattern structure framework is a generalization of FCA [5] where objects are described by *complex* representations. A pattern structure is a triple $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ where $\mathtt{G}$ is a set of objects, $(\mathtt{D}, \sqcap)$ is a semi-lattice of complex object representations, δ is a function that assigns to each object in $\mathtt{G}$ a representation in $\mathtt{D}$, and with $\mathtt{D}_\delta = \{d \in \mathtt{D} \mid \exists (X \subseteq \mathtt{G}) \sqcap_{g \in X} \delta(g) = d\}$, $(\mathtt{D}_\delta, \sqcap)$ is a complete subsemilattice of $(\mathtt{D}, \sqcap)$. The derivation operators for a pattern structure will be denoted as $(\cdot)^\square : \wp(\mathtt{G}) \to \mathtt{D}$ and $(\cdot)^\diamond : \mathtt{D} \to \wp(\mathtt{G})$. A pattern structure can be represented with a formal context, as the next proposition shows:

Proposition 1 ([5]). *Let $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ be a pattern structure, and let $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ be a formal context such that $\mathtt{M} \subseteq \mathtt{D}$ and $(g, m) \in \mathtt{I} \iff m \sqsubseteq \delta(g)$.*

If every element of $(\mathtt{D}_\delta, \sqcap)$ is of the form

$$\bigsqcup X = \bigsqcap \{d \in \mathtt{D}_\delta \mid (\forall x \in X) x \sqsubseteq d\}$$

*for some $X \subseteq \mathtt{M}$ (i.e. $\mathtt{M}$ is $\bigsqcup$-dense in $(\mathtt{D}_\delta, \sqcap)$), then $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ is a **representation context** (RC) of $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ and $(\mathtt{D}_\delta, \sqcap, \sqcup)$ is a complete lattice.*

The condition that $\mathtt{M}$ is $\bigsqcup$-dense in $(\mathtt{D}_\delta, \sqcap)$ means that any element in $\mathtt{D}_\delta$ can be represented (uniquely) as the *meet* of the filters of a set of descriptions $X \subseteq \mathtt{M}$ in $(\mathtt{D}_\delta, \sqcap)$. RCs yield concept lattices that are isomorphic to the pattern concept lattices of their corresponding pattern structures. For any pattern-intent $d \in \mathtt{D}_\delta$ in the pattern structure we have an intent $X \subseteq \mathtt{M}$ in the representation context such that $d = \bigsqcup X$ and $X = {\downarrow} d \cap \mathtt{M}$ where ${\downarrow} d$ is the ideal of d in $(\mathtt{D}, \sqcap)$.

The details on interval pattern structures can be found in [7], but we recall the following definitions. Given a many-valued context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, we have:

$$\begin{aligned}
\delta(g) &= \langle [\ m_i(g), m_i(g)\ ] \rangle_{i \in [1..|\mathtt{M}|]} \\
\delta(g_1) \sqcap \delta(g_2) &= \langle [\ min\{m_i(g_1), m_i(g_2)\}, max\{m_i(g_1), m_i(g_2)\}\ ] \rangle_{i \in [1..|\mathtt{M}|]} \\
A \subseteq \mathtt{G}\ ;\ A^\square &= \langle [\ min\{m_i(g) \mid g \in A\}, max\{m_i(g) \mid g \in A\}\ ] \rangle_{i \in [1..|\mathtt{M}|]} \\
d \in \mathtt{D}\ ;\ d^\diamond &= \{g \in \mathtt{G} \mid d \sqsubseteq \delta(g)\}
\end{aligned}$$

3 Building a Simple Representation Context

Algorithm 1 shows a first approach to build an RC for a pattern structure $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ with implementations for the derivation operators $(\cdot)^\square$ and $(\cdot)^\diamond$, given a many-valued context $\mathcal{M} = (\mathtt{G}, \mathtt{N}, \mathtt{W}, \mathtt{J})$. To distinguish the attributes in $\mathcal{M}$ from those in the RC created by Algorithm 1, we will refer to $\mathtt{N}$ as a set of *columns* in $\mathcal{M}$. Algorithm 1 is based on the NextClosure algorithm [3] for

calculating intents. Actually, it only differs in lines 12, 13, 14 (marked with an asterisk). Algorithm 1 starts by building the RC $\mathcal{K}$: the set of objects is the same as in the pattern structure, and the set of attributes and the incidence relation are initially empty. Line 12 checks whether a set of objects in the RC is an extent in the pattern structure. If this is the case, the algorithm continues executing NextClosure. Otherwise, the algorithm adds to the RC a new attribute corresponding to the pattern associated to the mismatching closure. It also adds to the incidence relation the pairs *object-attribute* as defined in line 14. Line 24 outputs the calculated RC. In what follows, we show that Algorithm 1 calculates a proper RC for the pattern structure defined over $\mathcal{M}$.

Proposition 2.
Algorithm 1 computes an RC $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ *of* $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$.

Proof. We show that $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ meets the conditions in Proposition 1. Similarly to NextClosure, Algorithm 1 enumerates all closures (given an arbitrary closure operator) in lectical order. However, Algorithm 1 uses two different closure operators, namely the standard closure operator of FCA defined over the RC under construction $(\cdot)''$, and the one defined by both derivation operators over the pattern structure, $(\cdot)^{\square\diamond}$. Both closure operators are made to coincide by the new instructions in the algorithm. When this is not the case (this is $B'' \neq B^{\square\diamond}$ for a given $B \subseteq \mathtt{G}$) a new attribute is added to the RC in the shape of $B^{\square}$. Additionally, the pair $(h, B^{\square})$ is added to the incidence relation set of the RC for all objects $h \in B$. This in turn ensures that $B'' = B^{\square\diamond}$ in the modified RC.

A consequence of $B'' = B^{\square\diamond}$ is that the set of extents in the RC is the same as the set of extents in the pattern structure. This also means that there is a one-to-one correspondence between the intents in the RC and the patterns in the pattern structure, i.e. for any extent $B'' = B^{\square\diamond}$, the intent $B''' = B'$ corresponds to the pattern $B^{\square\diamond\square} = B^{\square}$ ($B''' = B'$ and $B^{\square\diamond\square} = B^{\square}$ are properties of the derivation operators [4]). Thus, we have that any element in $\mathtt{D}_\delta$ (which can be represented as $B^{\square}$ for an arbitrary $B \subseteq \mathtt{G}$) is of the form $\bigsqcup B'$ or $\bigsqcap\{d \in \mathtt{D}_\delta \mid (\forall m \in B') m \sqsubseteq d\}$. Consequently, $\mathtt{M}$ is $\bigsqcup$-dense in $(\mathtt{D}_\delta, \sqcap)$ and $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ is an RC of the pattern structure $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$. □

We can consider the extreme case when the generated RC contains one attribute per pattern in the pattern structure to estimate the complexity of Algorithm 1. Taking from [3], we know that the polynomial delay of the base algorithm NextClosure is $\mathcal{O}(|\mathtt{G}|^2|\mathtt{M}|)$ (in here $\mathtt{G}$ and $\mathtt{M}$ are inverted as we are enumerating extents) which decomposes in the main loop of the algorithm (at most $|\mathtt{G}|$ repetitions), and the number of calculations taken by the closure operation in the formal context ($|\mathtt{G}||\mathtt{M}|$). Additionally, we need to consider the calculation of the closure operation over the pattern structure in line 12 of the algorithm which strongly depends on the nature of the pattern structure and its implementation. For example, for interval pattern structures, the closure takes $|\mathtt{G}||\mathtt{N}|$ operations. For such a case, the overall complexity is $\mathcal{O}(|\mathtt{G}|^2 \operatorname{argmax}(|\mathtt{N}|, |\mathtt{M}|))$.

Since we know $|\mathtt{N}|$, we need to characterize the size of $|\mathtt{M}|$. As previously stated, $|\mathtt{M}|$ may be as large as the size of patterns in $\mathtt{D}_\delta$, which would imply

Algorithm 1 Naive Representation Context Calculation

```
1: procedure RepresentationContextNaive(𝓜, (·)^□, (·)^◇)        ▷ 𝓜 = (G, N, W, J)
2:     M ← ∅
3:     I ← ∅
4:     𝒦 ← (G, M, I)
5:     A ← ∅
6:     while A ≠ G do
7:         for g ∈ G in reverse order do
8:             if g ∈ A then
9:                 A ← A \ {g}
10:            else
11:                B ← A ∪ {g}
12:                if B'' ≠ B^□◇ then                          ▷ (*)
13:                    M ← M ∪ {B^□}                           ▷ (*)
14:                    I ← I ∪ {(h, B^□) | h ∈ B^□◇}           ▷ (*)
15:                if B''\A contains no element < g then
16:                    A ← B''
17:                    Exit For
18:        Output A
19:    return 𝒦
20: end procedure
```

a non-polynomial delay of the algorithm (as $|\mathtt{D}_\delta|$ may grow up to $2^{|\mathtt{G}|}$). Differently, there may be cases when $|\mathtt{M}|$ is much smaller than $|\mathtt{D}_\delta|$. If by some happy accident the first patterns included in $\mathtt{M}$ correspond exactly to the set of join-irreducible patterns (JIPs) in $|\mathtt{D}_\delta|$, then we can expect that this would be the case. Particularly, when $|\mathtt{D}_\delta| = 2^{|\mathtt{G}|}$, we have that the number of JIPs in $\mathtt{D}_\delta$ is $|\mathtt{G}|$. Unfortunately, as stated in [4], determining the maximal number of JIPs for a given number of objects is difficult. An asymptotic upper bound is described in [2] as a factor of $\binom{|\mathtt{G}|}{\lfloor |\mathtt{G}|/2 \rfloor}$ which is still better than $2^{|\mathtt{G}|}$. Experience shows that for real-life datasets this number is much smaller, however to the best of our knowledge an actual study on this subject has not been performed yet.

Regardless, let us consider that by some clever mechanism Algorithm 1 is always able to add JIPs of the pattern structure to the RC. If this would be the case, any other pattern (this is, join-reducible pattern) could be represented as the join of a set of attributes in the RC (using Proposition 1), provided that these JIPs have been previously included in $\mathtt{M}$. The problem with this is that we cannot be certain when this last condition is met. This is, given a set of objects $B \subseteq \mathtt{M}$ and its closure in the RC B'', we cannot be sure whether B'' is an extent in the pattern structure until the algorithm has finished. The single exception is when $B = B''$ as we show in Proposition 3.

Proposition 3.
Let $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ *be the partial RC calculated by Algorithm 1 for the pattern structure* $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$*. Given a set of objects* $B \subseteq \mathtt{G}$*, at any point during the execution of Algorithm 1 we have that*

$$B = B'' \implies B = B^{\square\diamond}$$

Proof. Since a closure operator is extensive we have that $B \subseteq B''$ and $B \subseteq B^{\square\diamond}$. It suffices to show that $B \subseteq B^{\square\diamond} \subseteq B''$ at any point during the execution of

Algorithm 2 A General Representation Context Calculation

```
1: procedure RepresentationContext(𝓜, (·)^□, (·)^◇)                    ▷ 𝓜 = (G, N, W, J)
2:     M ← ∅
3:     I ← ∅
4:     𝒦 ← (G, M, I)
5:     A ← ∅
6:     while A ≠ G do
7:         for g ∈ G in reverse order do
8:             if g ∈ A then
9:                 A ← A \ {g}
10:            else
11:                B ← A ∪ {g}
12:                if B ≠ B'' then
13:                    while B'' ≠ B^□◇ do                              ▷ (*)
14:                        X ← Sample(B, B'', 𝓜)                        ▷ (*)
15:                        M ← M ∪ {m_X}          ▷ (m_X is a new attribute)
16:                        I ← I ∪ {(h, m_X) | h ∈ X}                   ▷ (*)
17:                if B''\A contains no element < g then
18:                    A ← B''
19:                    Exit For
20:        Output A
21:    return 𝒦
22: end procedure
```

Algorithm 1. This is a consequence of the lectical enumeration performed by the base algorithm NextClosure which ensures that whenever we calculate B'', we have already calculated and verified all closures $C'' = C^{\square\diamond}$ with $C'' \subseteq B$. Then, B'' can be characterized as $C'' \cap g''$ for some $C'' \subseteq B$ and $g \in \mathtt{G} \backslash C''$. Simultaneously, $B^{\square\diamond}$ can be characterized as $C^{\square\diamond} \cap \delta(g)^\diamond$ for the same set C and object g. Because $C'' = C^{\square\diamond}$, we need to show that $\delta(g)^\diamond \subseteq g''$. If we consider that $g' = \{X^\square \in \mathtt{M} \mid g \in X^{\square\diamond}\}$ (line 14 of Algorithm 1), it follows that $(\forall X^\square \in g') X^\square \sqsubseteq \delta(g) \implies \delta(g)^\diamond \subseteq X^{\square\diamond}$. Additionally, g'' can be also characterized as $\bigcap_{X^\square \in g'} X^{\square\diamond}$, then we have that $\delta(g)^\diamond \subseteq g''$. □

Proposition 3 tells us that for a given set $B \subseteq \mathtt{G}$ in the lectical enumeration, if we have that $B = B''$, it follows that this is true in the pattern structure without need of verification of $B^{\square\diamond}$ This is useful if the calculation of B'' in the RC is computationally cheaper than the calculation of $B^{\square\diamond}$ in the pattern structure. A corollary of Proposition 3 is that $g \notin B'' \implies g \notin B^{\square\diamond}$, meaning that B'' actually behaves as an estimation of $B^{\square\diamond}$ which gets refined the more attributes we add to the RC.

Our proposition provides a reduction in the complexity of calculating extents in a pattern structure whenever the size of $\mathtt{M}$ is smaller than the number of *columns* in the many-valued formal context (i.e. $|\mathtt{M}| < |\mathtt{N}|$). This is true when the algorithm begins execution. Moreover, the algorithm can keep track on this relation allowing for an adaptive strategy, i.e. falling back to NextClosure whenever the use of the proposed strategy becomes pointless.

Algorithm 3 An interval pattern extent sampler algorithm

```
1: procedure SAMPLE(B, B'', M)                                  ▷ M = (G, N, W, J)
2:     found ← False
3:     states ← list of size |N|
4:     for n ∈ N do
5:         side ← pick randomly from {0, 1}
6:         states[n] ← (side, 0, |W_n|)
7:     while not found do
8:         n ← pick randomly from N
9:         side, i, j ← states[n]
10:        a = i + side
11:        b = j + (side − 1)
12:        if a < b then
13:            X ← {g ∈ G | W_a^n ≤ n(g) ≤ W_b^n}
14:            side ← not side
15:            if B ⊆ X then
16:                i, j ← a, b
17:                if B'' ⊈ X then
18:                    found ← True
19:            states[n] = (side, i, j)
20:    return X
21: end procedure
```

4 Computing Smaller Representation Contexts

Algorithm 2 shows a better implementation of Algorithm 1 considering the results discussed in the previous section. It differs slightly from Algorithm 1 in those lines marked with an asterisk. Instead of adding a single attribute per set of objects whenever $B'' \neq B^{\square\diamond}$, it keeps on asking for *samples* until both closures coincide. Each *sample* corresponds to a set of objects $X \subseteq \mathtt{G}$ s.t. $X^{\square} \sqsubseteq B^{\square}$ and $(\exists g \in B'' \backslash B^{\square\diamond}) X^{\square} \not\sqsubseteq \delta(g)$. Notice that if there is no such $g \in B''$, then necessarily $B'' = B^{\square\diamond}$. Furthermore, when $X^{\square} = B^{\square}$, we fall back to Algorithm 1.

Ideally, $X^{\square}$ would be a join-irreducible pattern. However, we cannot be certain this is the case until we have calculated the entire set of patterns in the pattern structure which is exactly what we would like to avoid. For this reason, we simply require that the set X is as large as possible. This increases it chances that it may be a join-irreducible pattern.

Sampling strongly depends to the nature of the pattern structure. Algorithm 3 presents a simple sampling technique designed for interval pattern structures (IPS). We have chosen IPS to illustrate our approach for two reasons. Firstly, since they work directly on numerical data, they are prone to suffer from the curse of dimensionality. Secondly, because of their definition, IPS lattices are very large and their corresponding RCs usually contain contranominal scales.

For the sake of brevity we do not provide a full explanation on the inner working of Algorithm 3. We simply mention that, given a set of objects B and its *estimated* closure B'', it works by using the largest possible convex region in the description space and reducing it in one of its dimensions picked randomly. Objects within the reduced convex region are assigned to a set X. If $B \subseteq X$ and $B'' \not\subseteq X$, then X is retrieved as a sample (notice that a pre-condition of Algorithm 3 given by Algorithm 2 is that $B \neq B''$). Algorithm 3 provides an answer in $\mathcal{O}(|\mathtt{G}|^2|\mathtt{N}|)$.

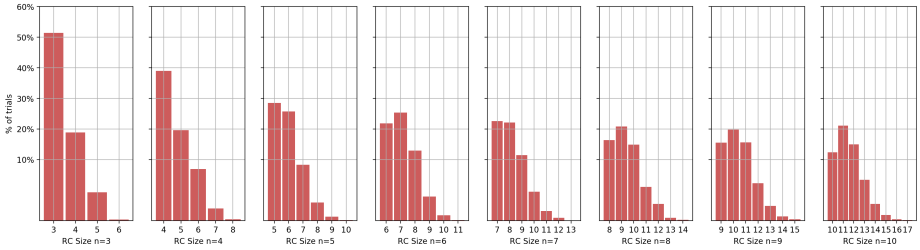

Fig. 1: Distribution on the number of attributes sampled for one thousand RCs. The x-axis has the number of attributes sampled. The y-axis represents the proportion of RCs in the total number of trials.

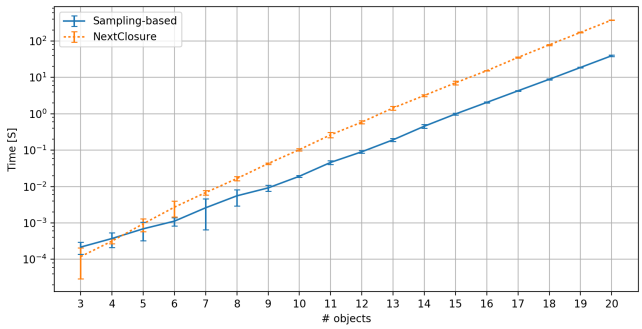

Fig. 2: Execution times for Algorithm 2 and 3 . The x-axis is the number of objects generated. The y-axis (time) is in logarithmic scale.

4.1 Experiments in Synthetic Data

To illustrate how our approach samples small RCs, we performed a simple experiment. We created several synthetic many-valued contexts with objects ranging from 2 to 10 such that an IPS defined over them would yield a Boolean lattice. An RC defined naively over these IPS would contain $2^{|\mathbb{G}|}$ attributes, however an RC containing only JIPs (a contranominal scale) would contain only $|\mathbb{G}|$ attributes. Fig. 1 shows the distribution on the number of attributes sampled over 1000 executions of Algorithms 2 and 3 over these datasets. From the figure, we can observe that very small RCs are sampled with very high probability. Moreover, for some many-valued context sizes, the RCs containing only JIPs are sampled with the highest probability. Figure2 shows a comparison on running times between standard NextClosure and our approach averaged over 100 runs over many-valued contexts with the same characteristics mentioned above, with a number of objects ranging from 3 to 20. While both techniques show an exponential growth in the running time w.r.t. the number of columns of the many-valued context (as expected), our approach shows a reduction in an order of magnitude over the NextClosure baseline.

5 Conclusions

We presented an approach to calculate pattern structure extents by means of sampling join-irreducible patterns to build small representation contexts. For this purpose, two algorithms were introduced for which we provided an analysis on their complexity. We have concluded that under some circumstances our approach may help to reduce the computational complexity of mining pattern structures. Initial evidence suggests that this is true for interval pattern structures, although more research is necessary in order to claim this is true also in real-world data.

Acknowledgments. This research work has been supported by the recognition of 2017SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya), and the grant TIN2017-89244-R from MINECO (Ministerio de Economía y Competitividad).

References

1. Aleksey Buzmakov. *Formal Concept Analysis and Pattern Structures for mining Structured Data*. PhD thesis, University of Lorraine, Nancy, France, 2015.
2. Paul Erdös and Daniel J. Kleitman. Extremal problems among subsets of a set. *Discrete Mathematics*, 306(10-11):923–931, 2006.
3. Benhard Ganter and Sergei Obiedkov. *Conceptual Exploration*. Springer, Berlin, 2016.
4. Benhard Ganter and Rudolph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.
5. Bernhard Ganter and Sergei O. Kuznetsov. Pattern Structures and Their Projections. In *Proceedings of ICCS 2001*, Lecture Notes in Computer Science 2120, pages 129–142. Springer, 2001.
6. Mehdi Kaytoue, Víctor Codocedo, Aleksey Buzmakov, Jaume Baixeries, Sergei O Kuznetsov, and Amedeo Napoli. Pattern Structures and Concept Lattices for Data Mining and Knowledge Processing. In *Proceedings of ECML-PKDD 2015 (Part III)*, volume 9286 of *Lecture Notes in Computer Science*, pages 227–231. Springer, 2015.
7. Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Revisiting Numerical Pattern Mining with Formal Concept Analysis. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1342–1347. IJCAI/AAAI, 2011.
8. Sergei O. Kuznetsov. Pattern Structures for Analyzing Complex Data. In Hiroshi Sakai, Mihir K. Chakraborty, Aboul Ella Hassanien, Dominik Slezak, and William Zhu, editors, *RSFDGrC*, volume 5908 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2009.