Building a Representation Context Based on Attribute Exploration Algorithms

Jaume Baixeries, Victor Codocedo, Mehdi Kaytoue, Amedeo Napoli

Building a Representation Context Based on Attribute Exploration Algorithms

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

[1] Computer Science Department. Universitat Politècnica de Catalunya. 08032, Barcelona. Catalonia.
[2] Departamento de Informática. Universidad Técnica Federico Santa María. Campus San Joaquín, Santiago de Chile.
[3] Infologic R&D, 99 avenue de Lyon, 26500 Bourg-Lès-Valence, France.
[4] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France.

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

Abstract. In this paper we discuss the problem of constructing the representation context of a pattern structure. First, we present a naive algorithm that computes the representation context of a pattern structure using an algorithm which is a variation of attribute exploration. Then, we study a different sampling technique for reducing the size of the representation context. Finally we show how to build such a reduced context and we discuss the possible ways of doing it.

1 Introduction

Formal Concept Analysis (FCA) has dealt with non binary data mainly in two different ways: one is by *scaling* [9] the original dataset into a formal context. This method has some drawbacks as it generates a formal context with larger dimensions than the original dataset and, depending on the method used to scale, some information may also be lost. Another way to deal with non binary data is by generalizing FCA into "pattern structures" [10,15], which allows handling complex data directly. Thus, instead of a analysing a binary relation between some objects and their attributes, it analyses a relation between objects and their representations, the values of which form a meet-semilattice. Pattern structures have proved useful for different unrelated tasks, such as, for instance to analyze gene expressions [14,13], compute database dependencies [1,2] or compute biclusters [6], or other structures data, like trees, intervals, graphs, sequences, fuzzy and heterogeneous data, among others [12].

Any pattern structure can be represented by an equivalent formal context, which is consequently called a "representation context" [10,4]. By *equivalent* we mean that there is a bijection between the intents in the representation context and the pattern-intents in the pattern structure. In fact, this means that both contain the same set of extents.

Pattern structures are difficult to calculate precisely because of the complex nature of the object representations they model. Often, pattern concept lattices

are very large and their associated sets of pattern concepts are difficult to handle. In this article we tackle this problem by mining a reduced "representation context" of pattern structures by means of *irreducible patterns* (IPs) [7,11]. IPs act as a basis –in the linear-algebraic sense– of the space of patterns, i.e. each pattern can be represented as a linear combination of the *join* of a subset of IPs. In the standard FCA notation, IPs correspond to the intents of $\bigwedge$-irreducible concepts. More specifically, we propose an algorithm that calculates the extents of a pattern structure while simultaneously calculating a small representation context. The later allows for a cheap calculation of most pattern extents and it is incrementally completed with *samples* obtained from the actual pattern structure. The algorithm is based on attribute exploration [8] (*object exploration* in this case) that instead of a domain-expert, uses an oracle to validate the question "is set X an extent in the pattern structure?". In the negative case, the oracle should be able to sample a new attribute to add in the representation context which is heuristically selected to be an IP (or close to it).

We first present a naive algorithm based on NextClosure [8] and attribute exploration techniques. Then, we present an alternative way to "call the oracle" with a sampling strategy which strongly depends on the nature of the object descriptions. We show that the sampling strategy is able to obtain very small representation contexts, and as well, it sometimes generates with the highest probability the representation contexts with irreducible attributes only.

This work mainly relies on studies about the formalization of representation contexts such as [10] and [4], and on the book [8]. Moreover, a short version of this paper was published in the proceedings of the ICFCA 2019 Conference [5].

2 Notation and Background

In the following we introduce some definitions needed for the development of this article. The notations used are based on [9]. 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 where $(g, m) \in \mathtt{I}$ denotes that "*object g has the attribute m*". The derivation operators are denoted as $' : \wp(\mathtt{G}) \to \wp(\mathtt{M})$ and $' : \wp(\mathtt{M}) \to \wp(\mathtt{G})$.

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. An example of a many-valued context can be found in Table 1 where $a(r_1) = 1$, $\mathtt{W}^b = \{1, 2\}$ and $\mathtt{W}^b_2 = 2$.

The pattern structure framework is a generalization of FCA [10] where objects may have *complex* descriptions. 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 descriptions, and δ is a function that assigns to each object in $\mathtt{G}$ a description in $\mathtt{D}$. The derivation operators for a pattern structure are denoted as $(\cdot)^{\square} : \mathtt{G} \to \mathtt{D}$ and

$(\cdot)^{\diamond} : \mathtt{D} \to \mathtt{G}$.

$$A \subseteq \mathtt{G}\ ;\ A^{\square} = \bigsqcap_{g \in A} \delta(g)$$
$$d \in \mathtt{D}\ ;\ d^{\diamond} = \{g \in \mathtt{G} \mid d \sqsubseteq \delta(g)\}$$

Pattern structures come in different flavors depending on the nature of the representation for which they are intended. Regardless of the nature of data, any pattern structure can be represented with a formal context, as the next definition shows:

Definition 1 ([10]). *Let* $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ *be a pattern structure, and let* $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ *be a context such that* $\mathtt{M} \subseteq \mathtt{D}$ *and* $(g, m) \in \mathtt{I} \iff m \sqsubseteq \delta(g)$.

If every intent of $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ *is of the form*

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

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

The condition that $\mathtt{M}$ is $\bigsqcup$-dense in $(\mathtt{D}, \sqcap)$ means that any element in $\mathtt{D}$ can be represented (uniquely) as the *meet* of the filters of a set of descriptions $X \subseteq \mathtt{M}$ in $(\mathtt{D}, \sqcap)$.

Representation contexts yield concept lattices that are isomorphic to the pattern concept lattices of their corresponding pattern structures. Moreover, there is a bijection between the intents in the representation context and the pattern-intents in the pattern structure. For any pattern-intent $d \in \mathtt{D}$ 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 $(\mathtt{D}, \sqcap)$.

3 Computing a Representation Context: a First and Naive Approach

3.1 The NaiveRepContext Algorithm

We first present a simple and naive method for building a representation context from a given pattern structure. Algorithm 1 shows this method in the form of a procedure called NaiveRepContext. This procedure is based on the standard NextClosure algorithm [8], to which some changes –marked with an asterisk– have been performed. These changes represent the interaction of the algorithm with the oracle.

The algorithm receives as inputs a copy of a many-valued context $\mathcal{M} = (\mathtt{G}, \mathtt{N}, \mathtt{W}, \mathtt{J})$ and implementations for the derivation operators $(\cdot)^{\square}$ and $(\cdot)^{\diamond}$ corresponding to the pattern structure $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$ defined over $\mathcal{M}$. To distinguish the attributes in the many-valued context from those in the representation context

created by Algorithm 1, we will refer to $\mathtt{N}$ as a set of *columns* in the many-valued context.

The algorithm starts by building the representation context $\mathcal{K}$. The set of objects is the same set of objects in the pattern structure, while the set of attributes and the incidence relation are initially empty. Line 12 checks whether a set of objects (B'') in the representation context is an extent in the pattern structure ($B^{\square\diamond}$). If this is the case, the algorithm continues executing NextClosure. Otherwise, the algorithm adds to the representation context a new attribute corresponding to the pattern associated to the mismatching closure. It also adds to the incidence relation the pairs *object-attribute*, i.e. $(h, B^{\square})$ as defined in line 14. Line 24 outputs the calculated representation context.

Algorithm 1 A Naive Calculation of the Representation Context.

```
1: procedure NaiveRepContext(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:                end if
16:                if B''\A contains no element < g then
17:                    A ← B''
18:                    Exit For
19:                end if
20:            end if
21:        end for
22:        Output A
23:    end while
24:    return K
25: end procedure
```

Proposition 1.
Algorithm 1 computes a representation context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ *of* $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$.

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

the incidence relation of the representation context for all objects $h \in B$. This in turn ensures that $B'' = B^{\Box\diamond}$ holds in the modified representation context.

A consequence of the equality $B'' = B^{\Box\diamond}$ is that the set of extents in the representation context 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 representation context and the patterns in the pattern structure, i.e. for any extent $B'' = B^{\Box\diamond}$, the intent $B''' = B'$ corresponds to the pattern $B^{\Box\diamond\Box} = B^{\Box}$ ($B''' = B'$ and $B^{\Box\diamond\Box} = B^{\Box}$ are properties of the derivation operators [9]). Thus, we have that any element in D, which can be represented as $B^{\Box}$ for an arbitrary $B \subseteq \mathtt{G}$, is of the form $\bigsqcup B'$ or $\bigsqcap\{d \in \mathtt{D} \mid (\forall m \in B')m \sqsubseteq d\}$. Consequently, $\mathtt{M}$ is $\bigsqcup$-dense in $(\mathtt{D}, \sqcap)$ and $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ is an RC of the pattern structure $(\mathtt{G}, (\mathtt{D}, \sqcap), \delta)$. □

3.2 An Example of Execution

Table 3 shows the execution trace of NaiveRepContext over an interval pattern structure defined over the many-valued formal context on Table 1. Columns in Table 3 are: row number, a candidate set B in Algorithm 1, its closure in the representation context, its closure in the pattern structure, the result of the test in line 12 of Algorithm 1, and the pattern-intent $B^{\Box}$. Notice that there are 26 entries in the last column of the table that correspond to the 26 different attributes in the resulting representation context in Table 4. The latter are enumerated in the order they appear in Table 3.

Examining the first row in Table 3, we observe that the algorithm initially calculates the closure of set $\{r_7\}$. At this point the representation context is empty so the closure of $\{r_7\}$ corresponds exactly to $\mathtt{G}$. However, using the pattern structure we find out that $\{r_7\}^{\Box\diamond} = \{r_7\}$ (test $B'' = B^{\Box\diamond}$ fails) and we need to add something to the representation context. Algorithms 1 adds to $\mathtt{M}$ the attribute corresponding to $\{r_7\}^{\Box}$ which is labelled as attribute d_1 in the representation context of Table 4 (thus ensuring that $\{r_7\}'' = \{r_7\}$)). The procedure is the same for the following sets in the lectic enumeration until we calculate the closure of $\{r_3\}$.

Two important things occur at this point: Firstly, $\{r_3\}^{\Box\diamond} = \{r_3\}$. Secondly, there is enough information in the representation context so that $\{r_3\}'' = \{r_3\}$ as well. Consequently, no new attribute is added to the representation context.

Let us discuss this example in more depth. The representation context at this point is conformed by the same elements in Table 4 truncated at column d_{10}. At this point, we should observe that $\{r_3\}^{\Box} = \bigsqcup\{r_3\}' = \bigsqcup\{d_6, d_8, d_9, d_{10}\}$.

Another important observation is that we do not really need to verify in the pattern structure that $\{r_3\}^{\Box\diamond} = \{r_3\}''$. Because closure is an extensive operation ($B \subseteq B''$ and $B \subseteq B^{\Box\diamond}$) at any point in the execution of Algorithm 1 we have that $B \subseteq B^{\Box\diamond} \subseteq B''$. Thus, $B = B'' \implies B = B^{\Box\diamond}$. This is indeed quite important since, as shown in Table 3, usually $B'' = B^{\Box\diamond}$ is true more often than not (in this example, 38 times out of 64). By avoiding the calculation of $B^{\Box\diamond}$ within the pattern structure we can avoid the costly calculation of pattern similarities and subsumptions by means of the representation context.

Table 1: Example Dataset 1

	a	b	c	d	e
r_1	1	1	1	1	1
r_2	2	1	1	1	1
r_3	3	1	2	2	2
r_4	3	2	3	2	2
r_5	3	1	2	3	2
r_6	1	1	2	2	2
r_7	1	1	2	4	2

Table 2: Example Dataset 2

	a	b	c	d
r_1	1	4	3	2
r_2	2	1	4	3
r_3	3	2	1	4
r_4	4	3	2	1

4 Computing Smaller Representation Contexts

4.1 Extending the NaiveRepContext Algorithm with Sampling

Algorithm 1 is able to calculate the representation context of the interval pattern structure created for the many-valued context in Table 1 rather inefficiently since it creates a different attribute for almost every interval pattern in the pattern structure. As previously described, Algorithm 1 is able to avoid creating attributes for some interval patterns when there is enough information in the partial representation context. More formally, it does not include a new attribute d in the representation context if and only if there exists a set $Y \subseteq \mathtt{M}$ such that $d = \bigsqcup Y$ or $d = \bigcap_{m \in Y} m^{\diamond}$. Knowing this, we are interested in examining whether there is a way to calculate a smaller representation context given an interval pattern structure definition.

Table 6 shows the execution trace of Algorithm 1 over the interval pattern structure defined over Table 2. We can observe that the algorithm generates 14 different attributes in the representation context, one for each interval pattern concept except for the top and bottom concepts. Notice that this pattern structure contains $2^4 = 16$ interval pattern concepts and since it has 4 objects, we can conclude that its associated concept lattice is Boolean.

This example is interesting because it is a worst-case scenario for Algorithm 1. In fact, it generated the largest possible representation context for the interval pattern structure derived from Table 2. Moreover, it verified each extent closure in the pattern structure. This example is even more interesting considering that for such an interval pattern structure there is a known *smallest* representation context which corresponds to a *contranominal scale* [8] that for this example would contain only 4 attributes as shown in Table 5.

To make matters worse, we can show that Algorithm 1 would behave the same for an interval pattern structure with an associated Boolean concept lattice of any size. Actually, this is a consequence of the lectic enumeration of object sets performed by Algorithm 1 which implies that whenever $B_0 \subseteq B_1$ (with B_0, B_1 closed sets in $\mathtt{G}$) then, B_0 is enumerated before B_1. Since a pattern-intent $B_1^{\square}$ is not included in the representation context only when there exists a set $Y \subseteq \mathtt{M}$ such that $\bigcap_{m \in Y} m^{\diamond} = B_1$ we have that $(\forall m \in Y) B_1 \subseteq m^{\diamond}$. Consequently, $B_1^{\square}$ would not be included in the representation context only when all $m \in X$ had been already enumerated which cannot be the case because of lectic numeration.

Table 3: Execution Trace of NaiveRepContext over Table 1

	B	B''	$B^{\square\diamond}$	$B'' = B^{\square\diamond}$	$B^{\square}$
1	r_7	G	r_7	$False$	$\langle[1,1],[1,1],[2,2],[4,4],[2,2]\rangle$
2	r_6	G	r_6	$False$	$\langle[1,1],[1,1],[2,2],[2,2],[2,2]\rangle$
3	r_6, r_7	G	r_6, r_7	$False$	$\langle[1,1],[1,1],[2,2],[2,4],[2,2]\rangle$
4	r_5	G	r_5	$False$	$\langle[3,3],[1,1],[2,2],[3,3],[2,2]\rangle$
5	r_5, r_7	G	r_5, r_7	$False$	$\langle[1,3],[1,1],[2,2],[3,4],[2,2]\rangle$
6	r_5, r_6	G	r_3, r_5, r_6	$False$	$\langle[1,3],[1,1],[2,2],[2,3],[2,2]\rangle$
7	r_4	G	r_4	$False$	$\langle[3,3],[2,2],[3,3],[2,2],[2,2]\rangle$
8	r_4, r_7	G	r_3, r_4, r_5, r_6, r_7	$False$	$\langle[1,3],[1,2],[2,3],[2,4],[2,2]\rangle$
9	r_4, r_6	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_6	$False$	$\langle[1,3],[1,2],[2,3],[2,2],[2,2]\rangle$
10	r_4, r_5	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5	$False$	$\langle[3,3],[1,2],[2,3],[2,3],[2,2]\rangle$
11	r_3	r_3	r_3	$True$	-
12	r_3, r_7	r_3, r_4, r_5, r_6, r_7	r_3, r_5, r_6, r_7	$False$	$\langle[1,3],[1,1],[2,2],[2,4],[2,2]\rangle$
13	r_3, r_6	r_3, r_6	r_3, r_6	$True$	-
14	r_3, r_6, r_7	r_3, r_5, r_6, r_7	r_3, r_5, r_6, r_7	$True$	-
15	r_3, r_5	r_3, r_5	r_3, r_5	$True$	-
16	r_3, r_5, r_7	r_3, r_5, r_6, r_7	r_3, r_5, r_6, r_7	$True$	-
17	r_3, r_5, r_6	r_3, r_5, r_6	r_3, r_5, r_6	$True$	-
18	r_3, r_5, r_6, r_7	r_3, r_5, r_6, r_7	r_3, r_5, r_6, r_7	$True$	-
19	r_3, r_4	r_3, r_4	r_3, r_4	$True$	-
20	r_3, r_4, r_7	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6, r_7	$True$	-
21	r_3, r_4, r_6	r_3, r_4, r_6	r_3, r_4, r_6	$True$	-
22	r_3, r_4, r_6, r_7	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6, r_7	$True$	-
23	r_3, r_4, r_5	r_3, r_4, r_5	r_3, r_4, r_5	$True$	-
24	r_3, r_4, r_5, r_7	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6, r_7	$True$	-
25	r_3, r_4, r_5, r_6	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6	$False$	$\langle[1,3],[1,2],[2,3],[2,3],[2,2]\rangle$
26	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6, r_7	r_3, r_4, r_5, r_6, r_7	$True$	-
27	r_2	G	r_2	$False$	$\langle[2,2],[1,1],[1,1],[1,1],[1,1]\rangle$
28	r_2, r_7	G	r_1, r_2, r_6, r_7	$False$	$\langle[1,2],[1,1],[1,2],[1,4],[1,2]\rangle$
29	r_2, r_6	r_1, r_2, r_6, r_7	r_1, r_2, r_6	$False$	$\langle[1,2],[1,1],[1,2],[1,2],[1,2]\rangle$
30	r_2, r_5	G	r_2, r_3, r_5	$False$	$\langle[2,3],[1,1],[1,2],[1,3],[1,2]\rangle$
31	r_2, r_4	G	r_2, r_3, r_4	$False$	$\langle[2,3],[1,2],[1,3],[1,2],[1,2]\rangle$
32	r_2, r_3	r_2, r_3	r_2, r_3	$True$	-
33	r_2, r_3, r_7	G	$r_1, r_2, r_3, r_5, r_6, r_7$	$False$	$\langle[1,3],[1,1],[1,2],[1,4],[1,2]\rangle$
34	r_2, r_3, r_6	$r_1, r_2, r_3, r_5, r_6, r_7$	r_1, r_2, r_3, r_6	$False$	$\langle[1,3],[1,1],[1,2],[1,2],[1,2]\rangle$
35	r_2, r_3, r_5	r_2, r_3, r_5	r_2, r_3, r_5	$True$	-
36	r_2, r_3, r_5, r_7	$r_1, r_2, r_3, r_5, r_6, r_7$	$r_1, r_2, r_3, r_5, r_6, r_7$	$True$	-
37	r_2, r_3, r_5, r_6	$r_1, r_2, r_3, r_5, r_6, r_7$	r_1, r_2, r_3, r_5, r_6	$False$	$\langle[1,3],[1,1],[1,2],[1,3],[1,2]\rangle$
38	r_2, r_3, r_4	r_2, r_3, r_4	r_2, r_3, r_4	$True$	-
39	r_2, r_3, r_4, r_7	G	G	$True$	-
40	r_2, r_3, r_4, r_6	G	r_1, r_2, r_3, r_4, r_6	$False$	$\langle[1,3],[1,2],[1,3],[1,2],[1,2]\rangle$
41	r_2, r_3, r_4, r_5	G	r_2, r_3, r_4, r_5	$False$	$\langle[2,3],[1,2],[1,3],[1,3],[1,2]\rangle$
42	r_2, r_3, r_4, r_5, r_7	G	G	$True$	-
43	r_2, r_3, r_4, r_5, r_6	G	$r_1, r_2, r_3, r_4, r_5, r_6$	$False$	$\langle[1,3],[1,2],[1,3],[1,3],[1,2]\rangle$
44	r_1	r_1, r_2, r_6	r_1	$False$	$\langle[1,1],[1,1],[1,1],[1,1],[1,1]\rangle$
45	r_1, r_7	r_1, r_2, r_6, r_7	r_1, r_6, r_7	$False$	$\langle[1,1],[1,1],[1,2],[1,4],[1,2]\rangle$
46	r_1, r_6	r_1, r_6	r_1, r_6	$True$	-
47	r_1, r_6, r_7	r_1, r_6, r_7	r_1, r_6, r_7	$True$	-
48	r_1, r_5	r_1, r_2, r_3, r_5, r_6	r_1, r_2, r_3, r_5, r_6	$True$	-
49	r_1, r_4	r_1, r_2, r_3, r_4, r_6	r_1, r_2, r_3, r_4, r_6	$True$	-
50	r_1, r_3	r_1, r_2, r_3, r_6	r_1, r_2, r_3, r_6	$True$	-
51	r_1, r_2	r_1, r_2, r_6	r_1, r_2	$False$	$\langle[1,2],[1,1],[1,1],[1,1],[1,1]\rangle$
52	r_1, r_2, r_7	r_1, r_2, r_6, r_7	r_1, r_2, r_6, r_7	$True$	-
53	r_1, r_2, r_6	r_1, r_2, r_6	r_1, r_2, r_6	$True$	-
54	r_1, r_2, r_6, r_7	r_1, r_2, r_6, r_7	r_1, r_2, r_6, r_7	$True$	-
55	r_1, r_2, r_5	r_1, r_2, r_3, r_5, r_6	r_1, r_2, r_3, r_5, r_6	$True$	-
56	r_1, r_2, r_4	r_1, r_2, r_3, r_4, r_6	r_1, r_2, r_3, r_4, r_6	$True$	-
57	r_1, r_2, r_3	r_1, r_2, r_3, r_6	r_1, r_2, r_3, r_6	$True$	-
58	r_1, r_2, r_3, r_6, r_7	$r_1, r_2, r_3, r_5, r_6, r_7$	$r_1, r_2, r_3, r_5, r_6, r_7$	$True$	-
59	r_1, r_2, r_3, r_5	r_1, r_2, r_3, r_5, r_6	r_1, r_2, r_3, r_5, r_6	$True$	-
60	$r_1, r_2, r_3, r_5, r_6, r_7$	$r_1, r_2, r_3, r_5, r_6, r_7$	$r_1, r_2, r_3, r_5, r_6, r_7$	$True$	-
61	r_1, r_2, r_3, r_4	r_1, r_2, r_3, r_4, r_6	r_1, r_2, r_3, r_4, r_6	$True$	-
62	$r_1, r_2, r_3, r_4, r_6, r_7$	G	G	$True$	-
63	r_1, r_2, r_3, r_4, r_5	$r_1, r_2, r_3, r_4, r_5, r_6$	$r_1, r_2, r_3, r_4, r_5, r_6$	$True$	-
64	G	G	G	$True$	-

Table 4: Representation Context of the Interval Pattern Structure of Table 1

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	d_{26}
r_1														×	×			×	×	×	×		×	×	×	×
r_2													×	×	×	×	×	×	×	×	×	×	×			×
r_3						×		×	×	×	×	×				×	×	×	×	×	×	×	×			
r_4							×	×	×	×		×					×				×	×	×			
r_5				×	×	×		×		×	×	×				×		×		×		×	×			
r_6		×	×			×		×	×		×	×		×	×			×	×	×	×		×		×	
r_7	×		×		×			×			×			×				×							×	

Table 5: Contranominal scale generated by Algorithm 3

	X_1	X_2	X_3	X_4
r_1		×	×	×
r_2	×		×	×
r_3	×	×		×
r_4	×	×	×	

Given a pattern structure with an associated Boolean concept lattice we would like that ideally, Algorithm 1 only adds irreducible attributes to the representation context. In this way we would maintain the $\bigsqcup$-dense in $(\mathtt{D}, \sqcap)$ property with a minimum number of attributes. Table 6 shows in gray those rows that represent the relevant attributes to add to the representation context. Adding only these four attributes would be enough to represent the set of interval pattern concepts. Nevertheless, algorithms that compute the irreducible attributes of a closure system have exponential complexity in the size of its formal context[3]. Instead, we resort to a sampling-based strategy to retrieve attributes with a large image in $\mathtt{G}$.

Table 6: Execution Trace of NaiveRepContext over Table 2

	B	B''	$B^{\square\diamond}$	$B'' = B^{\square\diamond}$	$B^{\square}$
1	r_4	r_1, r_2, r_3, r_4	r_4	$False$	$\langle[4,4],[3,3],[2,2],[1,1]\rangle$
2	r_3	r_1, r_2, r_3, r_4	r_3	$False$	$\langle[3,3],[2,2],[1,1],[4,4]\rangle$
3	r_3, r_4	r_1, r_2, r_3, r_4	r_3, r_4	$False$	$\langle[3,4],[2,3],[1,2],[1,4]\rangle$
4	r_2	r_1, r_2, r_3, r_4	r_2	$False$	$\langle[2,2],[1,1],[4,4],[3,3]\rangle$
5	r_2, r_4	r_1, r_2, r_3, r_4	r_2, r_4	$False$	$\langle[2,4],[1,3],[2,4],[1,3]\rangle$
6	r_2, r_3	r_1, r_2, r_3, r_4	r_2, r_3	$False$	$\langle[2,3],[1,2],[1,4],[3,4]\rangle$
7	r_2, r_3, r_4	r_1, r_2, r_3, r_4	r_2, r_3, r_4	$False$	$\langle[2,4],[1,3],[1,4],[1,4]\rangle$
8	r_1	r_1, r_2, r_3, r_4	r_1	$False$	$\langle[1,1],[4,4],[3,3],[2,2]\rangle$
9	r_1, r_4	r_1, r_2, r_3, r_4	r_1, r_4	$False$	$\langle[1,4],[3,4],[2,3],[1,2]\rangle$
10	r_1, r_3	r_1, r_2, r_3, r_4	r_1, r_3	$False$	$\langle[1,3],[2,4],[1,3],[2,4]\rangle$
11	r_1, r_3, r_4	r_1, r_2, r_3, r_4	r_1, r_3, r_4	$False$	$\langle[1,4],[2,4],[1,3],[1,4]\rangle$
12	r_1, r_2	r_1, r_2, r_3, r_4	r_1, r_2	$False$	$\langle[1,2],[1,4],[3,4],[2,3]\rangle$
13	r_1, r_2, r_4	r_1, r_2, r_3, r_4	r_1, r_2, r_4	$False$	$\langle[1,4],[1,4],[2,4],[1,3]\rangle$
14	r_1, r_2, r_3	r_1, r_2, r_3, r_4	r_1, r_2, r_3	$False$	$\langle[1,3],[1,4],[1,4],[2,4]\rangle$
15	r_1, r_2, r_3, r_4	r_1, r_2, r_3, r_4	r_1, r_2, r_3, r_4	$True$	-

4.2 Sampling Attributes for Interval Pattern Structures

Our sampling method is based on picking large convex regions in the space. To achieve this, we use a many-valued context denoted as $(\mathtt{G}, \mathtt{N}, \mathtt{W}, \mathtt{J})$ to differentiate it from the sets in the representation context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$. Given a set B and its closure in the representation context B'' –which we know to be different from $B^{\square\diamond}$– the goal of the sampling procedure is to find a set X such that $B \subseteq (X \cap B'')$. The sampling procedure starts by picking a random column in the many-valued context $n \in \mathtt{N}$. Then, we randomly pick a side to trim the ordered set $\mathtt{W}^n$ (*left* or *right*) to generate a new set $\hat{\mathtt{W}}^n$. A candidate set $X = \{g \in \mathtt{G} \mid n(g) \in \hat{\mathtt{W}}^n\}$ is created and we check whether $B \subseteq X$. If so, we return X if and only if $B'' \not\subseteq X$. In a different case, we pick a random column from $\mathtt{N}$ and proceed with the same instructions. This is a basic description of the sampling algorithm described in Algorithm 3. Some details on the creation of $\hat{\mathtt{W}}^n$ are left described only in the pseudocode.

Algorithm 3 is able to generate large convex regions by considering single dimensions of the space. It basically tries to answer the question: *Which is the largest region in any dimension which contains set B?* For example, let us try to sample an extent for the first row of Table 6 where $B = \{r_4\}$ and $B'' = \{r_1, r_2, r_3, r_4\}$. The many-valued context is showed in Table 2. Let us pick column b such that $W^b = \{1, 2, 3, 4\}$. Next, we pick *right* to create $\hat{\mathtt{W}}^b = \{1, 2, 3\}$. The candidate set is then $X_1 = \{r_2, r_3, r_4\}$ for which we have that $B \subseteq X_1$ and $B'' \not\subseteq X_1$ so we return $X_1 = \{r_2, r_3, r_4\}$.

We can observe that in the previous example $\{r_2, r_3, r_4\}$ corresponds to the image of an irreducible attribute in the representation context. Of course, this is a happy accident because we have made not-so-random decisions. True random decisions may lead to add reducible attributes into the representation context. However, since the random decisions are likely to pick large regions in the space, and thus attributes with a large image in $\mathtt{G}$, they are likely to be irreducible attributes.

Algorithm 2 adapts a sampling method into the generation of the representation context. Line 14 calls the sampling procedure such as the one described in Algorithm 3. Line 13 has been changed for a **while** instruction instead of an **if** instruction. This is because in this case the closure B'' should converge to $B^{\square\diamond}$ by the addition of one or more attributes into the representation context. This convergence is ensured since in the worst case scenario Algorithm 3 returns $B^{\square\diamond}$ as an attribute for the representation context.

Let us finish the previous example by using Algorithm 2. We notice that with $X_1 = \{r_2, r_3, r_4\}$ as the first object of the representation context we have that $\{r_4\}'' = \{r_2, r_3, r_4\}$ which is again different from $\{r_4\}^{\square\diamond}$. Consequently, Algorithm 2 calls for a new sample which by the same procedure described could be $X_2 = \{r_1, r_3, r_4\}$ (another happy accident). Notice that it cannot be again $\{r_2, r_3, r_4\}$ because of line 18 of Algorithm 3. Next, we have that $\{r_4\}'' = \{r_3, r_4\}$ calls for a new sample. This new sample could be $X_3 = \{r_1, r_2, r_4\}$ which renders $\{r_4\}'' = \{r_4\}^{\square\diamond}$. Algorithm 2 proceeds to calculate $\{r_3\}''$ which in the current state of the representation context yields $\{r_3, r_4\}$. If the SAMPLE procedure re-

Algorithm 2 A General Algorithm for Computing a Representation Context.

```
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:                    end while
18:                end if
19:                if B''\A contains no element < g then
20:                    A ← B''
21:                    Exit For
22:                end if
23:            end if
24:        end for
25:        Output A
26:    end while
27:    return 𝒦
28: end procedure
```

turns $X_4 = \{r_1, r_2, r_3\}$ we have then that $\{r_3\}'' = \{r_3\}^{\square\diamond}$. It should be noticed that at this point the representation context is complete w.r.t. the interval pattern structure. This is, any subsequent closure will be calculated in the representation context alone. Table 5 shows the contranominal scale corresponding to the representation context generated.

5 Conclusions

In this paper we have presented a sampling strategy in order to compute a smaller representation context for an interval pattern structure. We propose two algorithms to achieve such a computation, a first naive version based on object exploration, and a second improved version that uses a sampling oracle to quickly find irreducible patterns. These irreducible pattern can be considered the *basis* of a pattern structure. This paper is a first step towards the computation of minimal representation of a pattern structure by means of sampling techniques.

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. Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Computing Similarity Dependencies with Pattern Structures. In *Concept Lattices and their Applications*,

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:     end for
8:     while not found do
9:         n ← pick randomly from N
10:        side, i, j ← states[n]
11:        a = i + side
12:        b = j + (side − 1)
13:        if a < b then
14:            X ← {g ∈ G | W_a^n ≤ n(g) ≤ W_b^n}
15:            side ← not side
16:            if B ⊆ X then
17:                i, j ← a, b
18:                if B'' ⊈ X then
19:                    found ← True
20:                end if
21:            end if
22:            states[n] = (side, i, j)
23:        end if
24:    end while
25:    return X
26: end procedure
```


volume 1062, pages 33–44. CEUR Workshop Proceedings (ceur-ws.org), 2013.

2. Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterizing Functional Dependencies in Formal Concept Analysis with Pattern Structures. *Annals of Mathematics and Artificial Intelligence*, 72(1–2):129–149, 2014.
3. Alexandre Bazin. Comparing algorithms for computing lower covers of implication-closed sets. In *Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications, Moscow, Russia, July 18-22, 2016.*, pages 21–31, 2016.
4. Aleksey Buzmakov. *Formal Concept Analysis and Pattern Structures for mining Structured Data*. PhD thesis, University of Lorraine, Nancy, France, 2015.
5. Víctor Codocedo, Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Sampling Representation Contexts with Attribute Exploration. In Diana Cristea, Florence Le Ber, and Baris Sertkaya, editors, *Proceedings of the 15th International Conference on Formal Concept Analysis (ICFCA)*, Lecture Notes in Computer Science 11511, pages 307–314. Springer, 2019.
6. Víctor Codocedo and Amedeo Napoli. Lattice-based biclustering using Partition Pattern Structures. In *Proceedings of ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 213–218. IOS Press, 2014.
7. Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 1990.
8. Benhard Ganter and Sergei Obiedkov. *Conceptual Exploration*. Springer, Berlin, 2016.
9. Benhard Ganter and Rudolph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.
10. 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.

11. Michel Habib and Lhouari Nourine. Representation of lattices via set-colored posets. *Discrete Applied Mathematics*, 249:64–73, 2018.
12. 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.
13. 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.
14. Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. Mining gene expression data with pattern structures in Formal Concept Analysis. *Information Sciences*, 181(10):1989–2001, 2011.
15. 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.