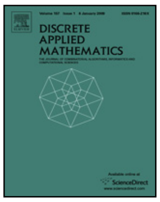
Discrete Applied Mathematics 159 (2011) 450–466

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam



On the complexity of enumerating pseudo-intents

Felix Distel [a], Barış Sertkaya [b,∗]

[a] *Theoretical Computer Science, TU Dresden, Nöthnitzer Str. 46 01187 Dresden, Germany*

[b] *SAP Research Center Dresden, Chemtnitzer Str. 48 01187 Dresden, Germany*

ARTICLE INFO

Article history:
Received 16 May 2010
Received in revised form 8 October 2010
Accepted 7 December 2010
Available online 12 January 2011

Keywords:
Formal concept analysis
Implications
Duquenne-Guigues Base
Pseudo-intents
Complexity

ABSTRACT

We investigate whether the pseudo-intents of a given formal context can efficiently be enumerated. We show that they cannot be enumerated in a specified lexicographic order with polynomial delay unless P = NP. Furthermore we show that if the restriction on the order of enumeration is removed, then the problem becomes at least as hard as enumerating minimal transversals of a given hypergraph. We introduce the notion of minimal pseudo-intents and show that recognizing minimal pseudo-intents is polynomial. Despite their less complicated nature, surprisingly it turns out that minimal pseudo-intents cannot be enumerated in output-polynomial time unless P = NP.

1. Introduction

Formal Concept Analysis (FCA) [16] is a field of applied mathematics with its roots in order theory, in particular the theory of complete lattices. FCA emerged in the 1980s from efforts to restructure lattice theory with the purpose of providing a lattice-theoretic formalization of the notions of a *concept* and a *conceptual hierarchy* [42,43]. Since then it has proven successful in various fields, including data analysis, machine learning and knowledge acquisition. Given a set of data consisting of objects and some of the properties of these objects, FCA builds an ordered set that reveals the inherent hierarchical structure and the implicit dependencies that occur between the properties of these objects.

In FCA, data is represented in a binary matrix called a *formal context*. A formal context is a simple way of specifying which *objects* have which *attributes*. Given a formal context, one way to analyze the data in this formal context is to compute (a canonical base of) the implications between its attributes. *Implications* between attributes are dependencies of the form "*every object that has the attributes* $m_{i1}, \ldots, m_{ik}$ *also has the attributes* $m_{j1}, \ldots, m_{jl}$". A formal context can have many implications, more precisely exponentially many in its size. Moreover, a large fraction of these implications are redundant, i.e., they can be derived from other implications. Therefore, one is interested in computing a non-redundant set of implications from which all implications of the formal context can be derived. In [20] Duquenne and Guigues give the definition of such an implicational base, which is called the *Duquenne–Guigues Base* of a formal context. The Duquenne–Guigues Base is not only non-redundant, but it is also of minimum cardinality, i.e., it contains the mininum number of implications that generate all implications. The Duquenne–Guigues Base of a formal context consists of the implications whose left hand sides are the so-called *pseudo-intents* of this formal context. Therefore, it is of crucial importance to efficiently enumerate pseudo-intents. Kuznetsov shows in [29] that the number of pseudo-intents can be exponential in the size of the formal context. Given this fact, it is clearly not possible to enumerate them in polynomial time. Therefore it makes sense to analyze this problem using the criteria introduced for measuring performance of enumeration

∗ Corresponding author. Tel.: +49 35148116225; fax: +49 62277853723.
E-mail addresses: felix@tcs.inf.tu-dresden.de (F. Distel), baris.sertkaya@sap.com (B. Sertkaya).

0166-218X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2010.12.004

Table 1
Formal context of quadrilaterals.

$\mathbb{K}_q$	Concave	Square	Rectangle	4 equal sides	Parallelogram
1					×
2					
3			×		×
4		×	×	×	×
5				×	×
6	×				

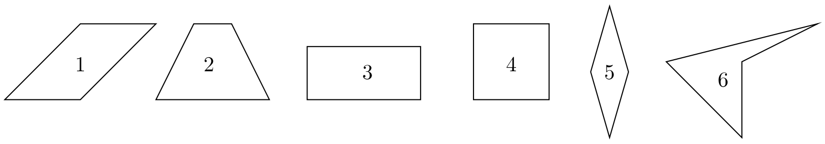


Fig. 1. Objects of the formal context of quadrilaterals.

problems [24]. Existing algorithms for enumerating pseudo-intents are not efficient w.r.t. these criteria. The most well-known algorithm *next-closure* introduced by Ganter in [14,15], besides pseudo-intents always generates the so-called concept intents as well, which can be exponentially more than the pseudo-intents. That is, the runtime of the algorithm is not bounded by a polynomial in the size of the output, i.e., it is not output-polynomial. Similarly, the time complexity of the *attribute incremental* algorithm introduced in [35] also depends on both the number of pseudo-intents and the number of concept intents. In the light of our current knowledge, it is not clear whether pseudo-intents can efficiently be enumerated.

The present work aims to answer this question. In Section 4, we investigate whether pseudo-intents can be enumerated in a specified lexicographic order with polynomial delay, i.e., with at most polynomial time between every pseudo-intent. It turns out that the first pseudo-intent can be computed in polynomial time, however, it is not possible to enumerate all of them with polynomial delay, unless P = NP. In Section 5, we remove the restriction on the order of enumeration and investigate whether it is possible to enumerate pseudo-intents in output polynomial time, i.e., time polynomial in the size of the input formal context and the number of pseudo-intents of this formal context. We show that in this setting enumerating pseudo-intents is at least as hard as enumerating the minimal transversals of a hypergraph, which is a prominent open problem. However, whether it is possible to enumerate them in output polynomial time or not, remains open. In Section 6 we restrict our attention to pseudo-intents that we call *minimal pseudo-intents*, which are pseudo-intents that do not contain another pseudo-intent. We show that minimal pseudo-intents can be recognized in polynomial time, however surprisingly they cannot be enumerated in output-polynomial time (unless P = NP) despite their less complicated nature. Some of our results have already appeared in [38,4].

2. Preliminaries

2.1. Formal concept analysis

In FCA, one represents data in the form of a *formal context*, which in its simplest form is a way of specifying which objects have which attributes:

Definition 1 (*Formal Context*)**.** A *formal context* is a triple $\mathbb{K} = (G, M, I)$, where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a relation that associates each object g with the attributes satisfied by g. In order to express that an object g is in relation I with an attribute m, we write gIm.

A formal context is visualized as a cross table, where the rows represent the objects, and the columns represent the attributes of the context. A cross in column m of row g means that object g has attribute m, and the absence of a cross means that g does not have attribute m. In the present work we consider only formal contexts with finite attribute sets. Example 2 demonstrates a formal context about quadrilaterals and some of their properties. In the following we are going to refer to a formal context only as a context for short.

Example 2. Table 1 demonstrates a context about quadrilaterals and their attributes *concave, square, rectangle*, 4 *equal sides*, and *parallelogram*. The objects of the context are depicted in Fig. 1. For instance the object 3 has the attributes *rectangle* and *parallelogram*, but it does not have the attributes *concave, square* and 4 *equal sides*.

Given a context, one can ask for the set of objects sharing a common set of attributes, or dually the set of attributes that are possessed by a common set of objects. These operations are called derivation operations.

Definition 3 (*Derivation Operator*)**.** Let $\mathbb{K} = (G, M, I)$ be a context. For a set of objects $A \subseteq G$, we define the set of attributes that all objects in A have in common as $A' := \{m \in M \mid \forall g \in A.\ gIm\}$. Similarly, for a set of attributes $B \subseteq M$, we define the set of objects that have all attributes in B as $B' := \{g \in G \mid \forall m \in B.\ gIm\}$.

For $A_1 \subseteq A_2 \subseteq G$ (resp. $B_1 \subseteq B_2 \subseteq M$), it is easy to see that

- $A_2' \subseteq A_1'$ (resp. $B_2' \subseteq B_1'$),
- $A_1 \subseteq A_1''$ and $A_1' = A_1'''$ (resp. $B_1 \subseteq B_1''$ and $B_1' = B_1'''$).

As an easy consequence of this, one obtains that double application of the derivation operator, i.e. $(\cdot)''$, yields a *closure operator* on both G and M. Using this closure operator one can describe a "natural clustering" between the objects and the attributes of a context.

Definition 4 (*Formal Concept*)**.** Let $\mathbb{K} = (G, M, I)$ be a context. A *formal concept* of $\mathbb{K}$ is a pair (A, B), where $A \subseteq G, B \subseteq M$ such that $A' = B$ and $B' = A$. A is called the *extent*, and B is called the *intent* of (A, B).

In the following, instead of formal concept, we will say concept for short. For every set $A \subseteq G$, A' is an intent of some concept since (A'', A') is always a concept. A'' is the smallest extent containing A. Consequently a set $A \subseteq G$ is an extent if and only if $A = A''$. The same applies to intents. The intersection of any number of extents (respectively intents) is always an extent (intent). Hence the set of all extents forms a closure system on G, and the set of all intents forms a closure system on M [16].

Given a set of data as a context, the most common method to analyze it is to find (a canonical base of) the implications between the attributes of this context. Implications between attributes are dependencies that hold in a given context.

Definition 5 (*Implication Between Attributes*)**.** Let $\mathbb{K} = (G, M, I)$ be a context. An *implication between the attributes* in M is a pair of sets $L, R \subseteq M$, written as $L \to R$. An implication $L \to R$ *holds* in $\mathbb{K}$ if every object of $\mathbb{K}$ that has all of the attributes in L also has all of the attributes in R, i.e., if $L' \subseteq R'$. We denote the set of all implications that hold in $\mathbb{K}$ by *Imp*$(\mathbb{K})$, and call it the *implicational theory* of $\mathbb{K}$.

An implication $L \to R$ holds in $\mathbb{K}$ iff R is contained in the $(\cdot)''$-closure of L, i.e., if $R \subseteq L''$. A set of implications induces its own closure operator, which is defined as follows:

Definition 6 (*Implicational Closure*)**.** Let $\mathcal{L}$ be a set of implications. For a set $P \subseteq M$, the *implicational closure* of P under $\mathcal{L}$, denoted by $\mathcal{L}(P)$, is the smallest subset Q of M such that $P \subseteq Q$, and $L \to R \in \mathcal{L}$ and $L \subseteq Q$ imply $R \subseteq Q$. It is easy to see that $\mathcal{L}(\cdot)$ is indeed a closure operator.

From the view point of logic, computing the implicational closure is just computing consequences in propositional Horn logic. In fact, the notions we have just defined can easily be reformulated in propositional logic. To this purpose, we view the attributes as propositional variables. An implication $L \to R$ can then be expressed by the formula $\phi_{L\to R} := \bigwedge_{r\in R}(\bigwedge_{\ell\in L} \ell \to r)$. Let $\Gamma_{\mathcal{L}}$ be the set of formulae corresponding to the set of implications $\mathcal{L}$. Then $\mathcal{L}(P) = \{b \in M \mid \Gamma_{\mathcal{L}} \cup \{\bigwedge_{p\in P} p\} \models b\}$, where $\models$ stands for classical propositional consequence. Since the formulae in $\Gamma_{\mathcal{L}}$ are Horn formulae, the implication closure $\mathcal{L}(B)$ can be computed in time linear in the sizes of $\mathcal{L}$ and B using methods for deciding satisfiability of sets of propositional Horn clauses [5]. Alternatively, these formulae can be viewed as expressing functional dependencies in a relational database, and thus the linearity result can also be obtained by using methods for deriving new functional dependencies from the given ones [33].

Definition 7 (*Implication Base*)**.** The implication $L \to R$ is said to *follow* from a set of implications $\mathcal{J}$ if $R \subseteq \mathcal{J}(L)$. The set of implications $\mathcal{J}$ is called *complete* for a set of implications $\mathcal{L}$ if every implication in $\mathcal{L}$ follows from $\mathcal{J}$. It is called *sound* for $\mathcal{L}$ if every implication that follows from $\mathcal{J}$ is contained in $\mathcal{L}$. A set of implications $\mathcal{J}$ is called a *base* for a set of implications $\mathcal{L}$ if it is both sound and complete for $\mathcal{L}$, and no strict subset of $\mathcal{J}$ satisfies this property.

For a given context $\mathbb{K} = (G, M, I)$, and a set of implications $\mathcal{J}$, if $\mathcal{J}$ is sound and complete for *Imp*$(\mathbb{K})$ then the two closure operators that we have introduced until now coincide, i.e., $B'' = \mathcal{J}(B)$ for all $B \subseteq M$. Consequently, given a base $\mathcal{J}$ for *Imp*$(\mathbb{K})$, any question of the form "$B_1 \to B_2 \in$ *Imp*$(\mathbb{K})$?" can be answered in time linear in the size of $\mathcal{J} \cup \{B_1 \to B_2\}$ since it is equivalent to asking whether $B_2 \subseteq B_1'' = \mathcal{J}(B_1)$.

The implicational theory of a context can be large. More precisely it can be exponentially large in the size of the given context [29]. Thus, one is interested in "small" bases generating the implicational theory. There may exist different implicational bases for a given context, not necessarily all of them with minimum cardinality. A base $\mathcal{J}$ of *Imp*$(\mathbb{K})$ is called a *minimum base* if no other base of *Imp*$(\mathbb{K})$ has cardinality less than the cardinality of $\mathcal{J}$. In [20] Duquenne and Guigues have defined such a base for contexts with a finite set of attributes. It is called the *Duquenne–Guigues Base* or the *stem base* of a context. The definition is based on the notion of a pseudo-intent:

Definition 8 (*Pseudo-Intent*)**.** A set $P \subseteq M$ is called a *pseudo-intent* of $\mathbb{K} = (G, M, I)$ if $P \neq P''$ and $Q'' \subsetneq P$ holds for every pseudo-intent $Q \subsetneq P$. Equivalently, a set $P \subseteq M$ is called a pseudo-intent if $P \neq P''$, it is a quasi-intent, and for every quasi-intent $Q \subsetneq P$, $Q'' \subsetneq P$ holds, where a *quasi-intent* is defined as a set $Q \subseteq M$ that satisfies $R'' \subseteq Q$ or $R'' = Q''$ for any $R \subseteq Q$.

The Duquenne–Guigues Base of a context $\mathbb{K}$ consists of the implications that have the pseudo-intents of $\mathbb{K}$ as left hand sides.

Definition 9 (*Duquenne–Guigues Base*)**.** The Duquenne–Guigues Base of a context $\mathbb{K}$ is the set of implications $\{P \to P'' \mid P$ a pseudo-intent of $\mathbb{K}\}$.

It has been shown in [20] that the Duquenne–Guigues Base is sound, complete and it is of minimum cardinality, i.e., it is a minimum base.

Example 10. For instance the Duquenne–Guigues-Base of the context $\mathbb{K}_q$ in Example 2 consists of the following 5 implications:

{concave, parallelogram} $\to$ {square, rectangle, 4 equal sides}
{square} $\to$ {rectangle, 4 equal sides, parallelogram}
{rectangle} $\to$ {parallelogram}
{4 equal sides} $\to$ {parallelogram}
{rectangle, 4 equal sides, parallelogram} $\to$ {square}.

The pseudo-intents of a context have the following property that we are often going to make use of in the coming proofs.

Proposition 11. *Let $\mathbb{K} = (G, M, I)$ be a formal context and $U \subseteq M$ a set of attributes. If U is not closed with respect to $(\cdot)''$ then there is a pseudo-intent $P \subseteq U$ such that $P'' \nsubseteq U$.*

Proof. In the case where there is a pseudo-intent $Q \subsetneq U$ such that $Q'' \nsubseteq U$ the claim holds trivially. We examine the remaining case where every pseudo-intent Q that is strictly contained in U satisfies $Q'' \subseteq U$. Since U is not closed with respect to $(\cdot)''$ it holds in particular that $U \neq Q''$ for every pseudo-intent $Q \subsetneq U$. Hence $Q'' \subsetneq U$ holds for every pseudo-intent $Q \subsetneq U$. Therefore U is itself a pseudo-intent by Definition 8. We have thus shown that U is a pseudo-intent that satisfies $U \subseteq U$ and $U'' \nsubseteq U$, i.e. the original claim holds for $P = U$. $\square$

2.2. Complexity of enumeration problems

As already mentioned in Section 1, the main problem that we consider in the present work is the problem of enumerating the pseudo-intents of a given context. For analyzing this problem we are often going to refer to notions from computational complexity theory [17,36], especially to notions about the complexity of enumeration problems which were introduced in [24].

In complexity theory we are sometimes interested not only in deciding whether a problem has a solution or not, but also in enumerating all solutions of this problem. We call such problems *enumeration problems*. For analyzing the complexity of enumeration problems where the number of solutions can be exponential in the size of the input, one needs appropriate measures. One such measure is the notion of *polynomial delay* [24]. We say that an algorithm that enumerates all solutions of an enumeration problem runs with polynomial delay if the time until the first solution is generated, and thereafter the time between every two consecutive solutions is bounded by a polynomial in the size of the input. An example of such an algorithm is the one given in [40] that enumerates all maximal independent sets of a graph with polynomial delay.

Another measure of performance for enumeration algorithms is to take into account not only the size of the input, but also the size of the output. We say that an algorithm runs in *output polynomial time* (or *polynomial total time*) [24] if it outputs all solutions in time polynomial in the size of the input *and the output*. Clearly, every polynomial delay algorithm is also an output polynomial algorithm, i.e., the notion of polynomial delay is stronger than the notion of output polynomial. An output polynomial algorithm runs in polynomial time (in the size of the input) if the problem has only polynomially many solutions.

A more complicated situation is when the solutions are required to be output in some prespecified order such as a lexicographic order. Obviously, this makes sense only in the case of polynomial delay; since if we are only interested in an output polynomial algorithm, then we can generate all solutions, sort them, and output them in the required order. A good example of such a polynomial delay algorithm is the one introduced in [24] that generates maximal independent sets of a graph in lexicographic order with polynomial delay.

2.3. Hypergraphs

In the coming sections while analyzing our problem of enumerating pseudo-intents, we are going to point out its close relation to a well-known enumeration problem in another field of mathematics, namely hypergraph theory. Therefore, we briefly recall basic notions from hypergraphs here. Hypergraph theory [2] is a field of discrete mathematics with many important applications in both theoretical and applied computer science. In its simplest form a hypergraph is a generalization of a graph, where edges can connect any number of vertices.

Definition 12. A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ is a pair consisting of a set of *vertices* $V = \{v_i \mid 1 \leq i \leq n\}$, and a set of *(hyper)edges* $\mathcal{E} = \{E_j \mid 1 \leq j \leq m\}$ where $E_j \subseteq V$.

Note that in the literature, e.g. [2], the edge set $\mathcal{E}$, as well as the edges E_j are sometimes required to be non-empty. Moreover, every node is required to occur in an edge, i.e., $\bigcup_{E\in\mathcal{E}} E = V$. For the sake of convenience with respect to the problems considered in the later sections, in the present work we adopt these requirements.

Definition 13. A set of vertices $W \subseteq V$ is called a *transversal* of $\mathcal{H}$ if it intersects every edge of $\mathcal{H}$, i.e., $\forall E \in \mathcal{E}.\, E \cap W \neq \emptyset$. A transversal is called *minimal* if no proper subset of it is a transversal. The set of all minimal transversals of $\mathcal{H}$ constitutes another hypergraph on V called the *transversal hypergraph* of $\mathcal{H}$, which is denoted by $\mathrm{Tr}(\mathcal{H})$.

Outside hypergraph theory, a hypergraph is just a collection of sets, a transversal is called a *hitting set*, and a minimal transversal is called a minimal hitting set. In graphs, a transversal is called a *vertex cover*, and a minimal transversal is called a minimal vertex cover. The minimal vertex covers of a graph can be efficiently computed. In [40,32] it has been shown that maximal independent sets of a graph, which are nothing but complements of minimal vertex covers, can be enumerated with polynomial delay. In [24] this result has been further improved, and it has been shown that maximal independent sets can be enumerated with polynomial delay even if they are required to be output in a specified lexicographic order. The problem is more challenging in the hypergraphs setting. Computing minimal transversals, i.e., generating $\mathrm{Tr}(\mathcal{H})$, is a prominent open problem:

Problem: TRANSVERSAL ENUMERATION (TRANS-ENUM)
Input: A hypergraph $\mathcal{H} = (V, \mathcal{E})$ on a finite set V.
Output: The edges of the transversal hypergraph $\mathrm{Tr}(\mathcal{H})$.
The well-known decision problem associated to this computation problem is defined as follows:

Problem: TRANSVERSAL HYPERGRAPH (TRANS-HYP)
Input: Two hypergraphs $\mathcal{H} = (V, \mathcal{E}_{\mathcal{H}})$ and $\mathcal{G} = (V, \mathcal{E}_{\mathcal{G}})$.
Question: Is $\mathcal{G}$ the transversal hypergraph of $\mathcal{H}$, i.e., does $\mathrm{Tr}(\mathcal{H}) = \mathcal{G}$ hold?

Computational complexity of this problem has now been extensively studied [7,8,13,10,27,26,11,12], and many important problems from various fields of computer science have been shown to be computationally equivalent to this problem. Some of these problems are: from relational databases the problem FD-RELATION EQUIVALENCE, which is checking whether a given set of functional dependencies that is in Boyce-Codd Normal Form is a cover of a given relation instance [8], the problem ADDITIONAL KEY for relation instances, which is the problem of checking whether an additional key exists for a given relation instance and a set of minimal keys thereof [8], and from logic the problem MONOTONE BOOLEAN DUALITY, which is checking whether two monotone Boolean functions given by their irredundant disjunctive normal forms are mutually dual [13]. Other equivalent problems from artificial intelligence can be found in [9,25], problems from data mining can be found in [21], and a comprehensive survey on related problems from various fields of computer science can be found in [22]. TRANS-HYP is known to be in CONP, however whether it is CONP-hard or whether it is solvable in polynomial time has now been open for more than 20 years. Similarly, it is an open problem whether TRANS-ENUM can be solved in output-polynomial time. It is known that if TRANS-HYP turns out to be CONP-complete then, unless P = NP, TRANS-ENUM cannot be solved in output-polynomial time. In a landmark paper [13] Fredman and Khachiyan proved that MONOTONE BOOLEAN DUALITY, and thus TRANS-HYP, can be solved in $n^{o(\log n)}$ time, which implies that these problems are most likely not CONP-hard.

In the following we say that a decision problem Π is TRANS-HYP-hard if TRANS-HYP can be reduced to Π by a standard polynomial transformation. We say that Π is TRANS-HYP-complete if it is TRANS-HYP-hard and Π can be reduced to TRANS-HYP by a polynomial transformation.

3. Related work and previous results

Pseudo-intents and computational problems related to them have attracted major interest in the FCA community [14,6,41,29,30,23,37,35,31,38,39,15,1] since their introduction in [20]. Due to the key role they play in FCA, it is of crucial importance to enumerate pseudo-intents efficiently. It is well known that the number of pseudo-intents of a context $\mathbb{K} = (G, M, I)$ can be exponential in the size of the attribute set, i.e., $|M|$. This is for instance the case when object intents are precisely all possible subsets of M with cardinality $|M|/2$. However, in this case both the number of objects, i.e., $|G|$, and the size of the incidence relation, i.e., $|I|$ are also exponential in $|M|$. Thus the number of pseudo-intents is polynomial in $|I|$. For a long time it was not known whether the number of pseudo-intents can also be exponential in $|I|$. In [29] Kuznetsov has shown that this can be the case. He has given an example of a context where the number of pseudo-intents is exponential in the size of the incidence relation. Moreover, he has also shown that determining the number of pseudo-intents is a #P-hard problem, i.e., it is intractable. Given the fact that the number of pseudo-intents can be exponential in the size of the context, it is clearly not possible to enumerate all pseudo-intents in time polynomial in the size of the input context. Therefore it makes sense to analyze the problem using the measures introduced in Section 2.2.

For enumerating pseudo-intents the most well-known algorithm is the *next-closure* algorithm [14,15] by Ganter. Originally, the algorithm was designed to enumerate all closed sets of a given closure operator. It is not specifically tailored for enumerating pseudo-intents. Given a closure operator, it enumerates all closed sets in a particular lexicographic order called the *lectic order*:

Table 2
Context where the number of intents is exponential in the number of pseudo-intents.

$\mathbb{K}_4$	1	2	3	4
g_1	X	X		
g_2	X		X	
g_3	X			X
g_4		X	X	
g_5		X		X
g_6			X	X

Definition 14 (*Lectic Order*)**.** Let $M = \{m_1, \dots, m_n\}$ and fix some linear order $m_1 < m_2 < \cdots < m_n$ on M. This order imposes a linear order on the power set of M, called the *lectic order*, which is also denoted by $<$, and is defined as:

$$A < B \quad \text{iff } \exists m_i \in B \setminus A : A \cap \{m_1, \dots, m_{i-1}\} = B \cap \{m_1, \dots, m_{i-1}\}. \quad \square$$

Obviously, $<$ extends the strict subset order, i.e., $A \subsetneq B$ implies $A < B$, and thus $\emptyset$ is the smallest and M is the largest set w.r.t. $<$.

In Section 2.1 we have mentioned that the set of all intents of a context forms a closure system on M, and the set of all extents forms a closure system on G. Given the closure operator $(\cdot)''$ on the attribute set M (respectively on the object set G) of a context $\mathbb{K} = (G, M, I)$ the next-closure algorithm enumerates all concept intents (respectively extents) of $\mathbb{K}$ in the lectic order. Moreover, the algorithm is efficient since it has polynomial delay in the size of $\mathbb{K}$.

In addition to introducing the next-closure algorithm, in [14,15] Ganter has also shown that the intents of a context together with its pseudo-intents form another closure system. By using this property and the next-closure algorithm, one can easily enumerate the intents and the pseudo-intents of a context. The only thing needed is the closure operator for this new closure system. In order to find the pseudo-intents, whenever the algorithm generates a new closed set, one checks whether it is an intent. If this is not the case, then it is a pseudo-intent. Recall that checking whether a given set $A \subseteq M$ is an intent can easily be done in polynomial time by checking whether $A = A''$ holds. This way we can enumerate all pseudo-intents. However, note that this is not an efficient method for enumerating pseudo-intents. It is well-known that a context can have exponentially many intents. Thus, for such a context, the algorithm might have to generate exponentially many intents between two consecutive pseudo-intents, i.e., it is not polynomial delay. Even worse, as seen in Example 15, the number of intents of a context can be exponential in the number of its pseudo-intents. Thus the runtime of the algorithm is not bounded by a polynomial in the number of pseudo-intents, i.e., it is not even output-polynomial.

Example 15. Consider a context $\mathbb{K}_n = (G_n, M_n, I_n)$, where $M_n = \{1, \dots, n\}$ and all subsets of M_n with cardinality $n-2$ are object intents. Thus the context has $n(n-1)/2$ objects, and n attributes. The pseudo-intents of $\mathbb{K}_n$ are exactly those sets of cardinality $n-1$, because they are not closed and all sets of cardinality less than $n-1$ are closed i.e., they are intents. That is there are $2^n - n - 1$ intents, while there are only n pseudo-intents. The case for $n = 4$ is shown in Table 2. $\square$

One other well-known algorithm for enumerating pseudo-intents is the *attribute-incremental* algorithm introduced by Obiedkov and Duquenne in [35]. As in the case of the next-closure algorithm, the runtime of this algorithm depends not only on the number of pseudo-intents, but also on the number of intents. That is this algorithm is not output-polynomial either.

A number of special cases of the problem have been considered in the literature as well. In [23] Janssen and Nourine have shown that for the case where the concept lattice is meet-semidistributive, there are at most polynomially many pseudo-intents, and they can be enumerated in polynomial time. For the case where the concept lattice is modular, Wild has shown in [41] that an optimal base, i.e., a base that not only contains the minimum number of implications, but also contains the minimum number of attributes, can be computed in polynomial time. In [6] Duquenne has shown that for locally distributive lattices a minimum cardinality base can be computed in polynomial time.

Apart from enumerating them, various other computational aspects of pseudo-intents have been considered in the literature. In [18] Gély et al. have investigated possible reasons why the number of pseudo-intents can be exponential in the size of the given context. They have shown that in some cases the exponential blow-up arises from the so-called *P-clone* attributes, which are attributes that can be exchanged to give new pseudo-intents. Moreover, they have introduced an operation that can be used to have a compact representation of pseudo-intents by using the P-clone attributes. In [34] Medina et al. have presented efficient algorithms for detecting whether two given attributes are clone attributes. In [19] Gély and Nourine have worked on closure systems that have the same set of non-unit pseudo-intents, i.e., pseudo-intents that have cardinality greater than one. They have given a polynomial algorithm that computes the meet-irreducible elements of a minimal closure system that preserves the implications whose left hand sides are pseudo-intents of this type. In [3] Colomb and Nourine have pointed out the relation between k-conformal hypergraphs and the keys of a context, where a key means a set of attributes that is not contained in any object intent. They have also shown that the problem of deciding whether a key of size larger than a given number is NP-complete.

One other important problem on pseudo-intents is of course recognizing them, i.e., deciding whether a given set is a pseudo-intent of a given context. In [30,31] Kuznetsov and Obiedkov have shown that this problem is in coNP. The lower

Algorithm 1 Algorithm for finding the lectically first pseudo-intent

```
1: Input: $\mathbb{K} = (G, M, I)$, linear order $<$ on $M$ s.t. $m_1 < \cdots < m_n$
2: $S_0 := M, n := |M|$
3: for $i = 1$ to $n$ do
4:   if $S_{i-1} \setminus \{m_i\}$ contains a pseudo-intent then
5:     $S_i = S_{i-1} \setminus \{m_i\}$
6:   else
7:     $S_i = S_{i-1}$
8:   end if
9: end for
10: if $S_n = M$ then
11:   print $\mathbb{K}$ has no pseudo-intents
12: else
13:   return $S_n$
14: end if
```

bound of this problem has been open for a long time, i.e., it was not known whether it is coNP-hard, or polynomial time solvable. Recently in [1] Babin and Kuznetsov have shown that it is coNP-hard.[1] In [37] Rudolph has presented a worst case exponential algorithm for this problem, and given an optimization based on reduced contexts.

4. Complexity of enumerating pseudo-intents in a specified order

In the present section we investigate whether it is possible to efficiently enumerate pseudo-intents in the lectic order with polynomial delay as in the case of intents. Clearly, in order to be able to do this, we need to be able to compute the lectically first pseudo-intent in polynomial time. We start with analyzing the complexity of this problem.

4.1. Complexity of computing the lectically first pseudo-intent

Our problem is formally defined as follows:

Problem: LECTICALLY FIRST PSEUDO-INTENT (FIRST-PI)
Input: A formal context $\mathbb{K} = (G, M, I)$, a linear order on M.
Output: The pseudo-intent that is the first one w.r.t. the lectic order induced by the given linear order.
As we shall see next, FIRST-PI can be solved in polynomial time. Our algorithm is based on the following property:

Lemma 16. *Let $S \subseteq M$ be a set of attributes. S contains a pseudo-intent iff S or one of the sets $S \setminus \{m\}$, where $m \in S$, is not closed.*

Proof. ($\Rightarrow$) Assume that all the sets $S \setminus \{m\},\ m \in S$, and S itself are closed. All strict subsets of S can be written as the intersection of sets of the form $S \setminus \{m\}$. The intersection of closed sets is closed and therefore no strict subset of S can be a pseudo-intent. Since S is closed S cannot be a pseudo-intent itself. This contradicts the assumption that S contains a pseudo-intent. Therefore, at least one of the sets S or $S \setminus \{m\},\ m \in S$, must be closed.

($\Leftarrow$) Let U be a set of attributes that is not closed. Proposition 11 shows in particular that every set that is not closed contains a pseudo-intent. Hence, S must contain a pseudo-intent. □

Recall that checking whether a set is closed can be performed in $\mathcal{O}(|G|\,|M|)$ time. Therefore checking whether S contains a pseudo-intent can be done in $\mathcal{O}(|G|\,|M|\,|S|)$ time. Algorithm 1 is based on Lemma 16.

Lemma 17 (*Correctness of Algorithm 1*)**.** *Let $\mathbb{K} = (G, M, I)$ and a linear order $m_1 < \cdots < m_n$ on M be given. Algorithm 1 terminates after a polynomial number of steps, and upon termination returns lectically the first pseudo-intent of $\mathbb{K}$ if it has any.*

Proof. Termination is clear since n is finite. Let P be lectically the first pseudo-intent. We show by induction that after the i-th iteration of the **for**-loop it holds that $P \cap \{m_1, \ldots, m_i\} = S_i \cap \{m_1, \ldots, m_i\}$ and $P \subseteq S_i$. This statement is true for $i = 0$ ($P \cap \emptyset = S_0 \cap \emptyset$). Assume that $P \cap \{m_1, \ldots, m_{i-1}\} = S_{i-1} \cap \{m_1, \ldots, m_{i-1}\}$ and $P \subseteq S_{i-1}$. We need to show that $m_i \notin P$ if and only if $S_{i-1} \setminus \{m_i\}$ contains a pseudo-intent.

($\Leftarrow$) Let Q be a pseudo-intent that is contained in $S_{i-1} \setminus \{m_i\}$. If $P = Q$, obviously P does not contain m_i. If $P < Q$, the smallest attribute that distinguishes P and Q must be in Q. Thus P cannot contain m_i.

[1] In [39] Sertkaya has given a proof that this problem is TRANS-HYP-hard. However, later it turned out that there was a mistake in this proof. The second author would like to thank Mikhail A. Babin for pointing out this error.

($\Rightarrow$) If $S_{i-1} \setminus \{m_i\}$ does not contain a pseudo-intent then in particular $P \not\subseteq S_{i-1} \setminus \{m_i\}$. Since $P \subseteq S_{i-1}$, it follows that $m_i \in P$. Therefore upon termination, i.e. after the n-th iteration of the **for**-loop it holds that $P = P \cap M = S_n \cap M = S_n$.

The **for**-loop iterates at most n times. Since checking whether a set $S_{i-1} \setminus \{m_i\}$ contains a pseudo-intent can be done in $\mathcal{O}(|G|\,|M|\,|S_{i-1} \setminus \{m_i\}|)$ time, the whole algorithm terminates in $\mathcal{O}(|G|\,|M|^3)$ time. □

The result is encouraging, but as we shall see next in Section 4.2 unfortunately it does not mean that pseudo-intents can efficiently be enumerated in the lectic order. After computing the first one, we cannot efficiently compute the subsequent ones.

4.2. Complexity of enumeration in the lectic order

In the present section we investigate the complexity of computing the subsequent pseudo-intents after computing lectically the first one. In order to analyze this computation problem, we consider the following decision problem.
Problem: Lectically first-n pseudo-intents (first-n-pi)

Input: A formal context $\mathbb{K} = (G, M, I)$ and pseudo-intents $P_1, \ldots, P_n$.

Question: Are $P_1, \ldots, P_n$ lectically the first n pseudo-intents of $\mathbb{K}$?

As we shall see later, if this problem is not decidable in polynomial time, then pseudo-intents cannot be enumerated in the lectic order with polynomial delay. We first show that this problem is in coNP.

Proposition 18. *$P_1, \ldots, P_n$ are not lectically the first n pseudo-intents of $\mathbb{K}$ iff there is a set $Q \subseteq M$ such that*

1. *Q is lectically smaller than P_j for some $j \in \{1, \ldots, n\}$, and*
2. *Q is not closed, and*
3. *for all $i \in \{1, \ldots, n\}$ either $P_i \not\subseteq Q$ or $P_i'' \subseteq Q$.*

Proof. ($\Leftarrow$) Since Q is not closed, by Proposition 11 there is a pseudo-intent P of $\mathbb{K}$ such that $P \subseteq Q$ but $P'' \not\subseteq Q$. Because of 3 it holds that $P \notin \{P_1, \ldots, P_n\}$. P is lectically smaller than P_j because Q is lectically smaller than P_j and $P \subseteq Q$. Thus $P_1, \ldots, P_n$ are not the lectically smallest pseudo-intents of $\mathbb{K}$.

($\Rightarrow$) Let P be a pseudo-intent that is lectically smaller than P_j, for some $j \in \{1, \ldots, n\}$ but not contained in $\{P_1, \ldots, P_n\}$. Then $Q = P$ satisfies the three conditions 1–3. □

Lemma 19 (*Containment in* coNP)**.** first-n-pi *is in* coNP.

Proof. Whether a set $Q \subseteq M$ satisfies conditions 1–3 in Proposition 18 can be checked in time polynomial in the size of $\mathbb{K}$ and $P_1, \ldots, P_n$. In order to decide whether $P_1, \ldots, P_n$ are not lectically the first n pseudo-intents of $\mathbb{K}$ one can non-deterministically guess a subset $Q \subseteq M$ and then check in polynomial time whether it satisfies the conditions 1–3. Hence the dual problem is in NP and thus first-n-pi is in coNP. □

Next we show that first-n-pi is coNP-hard. For showing hardness, we give a reduction from the well-known coNP-hard problem validity [36]. The validity problem asks whether a given Boolean formula in DNF is valid, i.e., it evaluates to *true* for every truth assignment. Recall that validity is closely related to satisfiability, which is the problem of checking whether a given CNF Boolean formula is satisfiable. A CNF formula φ is unsatisfiable if and only if $\neg\varphi$ is valid, which implies that validity is coNP-hard.

Problem: validity

Input: A Boolean formula f given in DNF.

Question: Is f valid?

In order to prove that first-n-pi is at least as hard as validity, we use the following construction: Let an instance of validity be given with the DNF formula $f(p_1, \ldots, p_m) = D_1 \vee \cdots \vee D_k$, where $D_i = (x_{i1} \wedge \cdots \wedge x_{il_i})$ and $x_{ir} \in \{p_1, \ldots, p_m\} \cup \{\neg p_1, \ldots, \neg p_m\}$ for all $i \in \{1, \ldots, k\}$ and all $r \in \{1, \ldots, l_i\}$. From f we construct a formal context $\mathbb{K}_f = (G, M, I)$ as follows: We define the set of attributes $M = \{\alpha_1, \ldots, \alpha_m, t_1, \ldots, t_m, f_1, \ldots, f_m\}$, and order the elements of M as $\alpha_1 < \cdots < \alpha_m < t_1 < f_1 < \cdots < t_m < f_m$.

In order to define the object intents, for every $i \in \{1, \ldots, k\}$ we define a set

$$A_i = M \setminus \{f_j \mid p_j \text{ occurs in } D_i \text{ as a positive literal}\} \setminus \{t_j \mid p_j \text{ occurs in } D_i \text{ as a negative literal}\} \setminus \{\alpha_j \mid p_j \text{ occurs in } D_i\}.$$

Furthermore for every $i \in \{1, \ldots, k\}$ and every $j \in \{1, \ldots, m\}$ we define two sets F_{ij} and T_{ij} as $T_{ij} = A_i \setminus \{f_j, \alpha_j\}$, and $F_{ij} = A_i \setminus \{t_j, \alpha_j\}$. Using these we define the set of objects as $G = \{u_1, \ldots, u_{2m}\} \cup \{g_{T_{ij}} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}\} \cup \{g_{F_{ij}} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}\}$. The relation I is defined so that every object $g_{T_{ij}}$ has all the attributes that are contained in the set T_{ij} and analogously for $g_{F_{ij}}$. Every singleton set $\{t_j\}$ or $\{f_j\}$ occurs as the intent of some object u_r. More formally, we define:

$$\begin{aligned} I = {} & \{(u_{2j-1}, t_j) \mid j \in \{1, \ldots, m\}\} \cup \{(u_{2j}, f_j) \mid j \in \{1, \ldots, m\}\} \\ & \cup \{(g_{F_{ij}}, x) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}, x \in F_{ij}\} \cup \{(g_{T_{ij}}, x) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}, x \in T_{ij}\}. \end{aligned}$$

Table 3 demonstrates the context $\mathbb{K}_f$. Note that $\mathbb{K}_f$ has $2mk + 2m$ objects and $3m$ attributes, so its size is $\mathcal{O}(m^2k + m^2)$. Finally we define the sets $P_1, \ldots, P_m$ as: $P_j = \{t_j, f_j\}$ for $j \in \{1, \ldots, m\}$.

Table 3
Context $\mathbb{K}_f$ constructed from the Boolean DNF formula f.

$\mathbb{K}_f$	α_1	$\cdots$	α_m	t_1	f_1	t_2	f_2	$\cdots$	t_m	f_m
u_1				X						
$\vdots$							$\cdots$			
u_{2m}										X
$g_{T_{11}}$			$\cdots$		T_{11}		$\cdots$			
$\vdots$					$\vdots$					
$g_{T_{km}}$			$\cdots$		T_{km}		$\cdots$			
$g_{F_{11}}$			$\cdots$		F_{11}		$\cdots$			
$\vdots$					$\vdots$					
$g_{F_{km}}$			$\cdots$		F_{km}		$\cdots$			

The construction may look complicated at first glance. The basic ideas underlying the construction are as follows:

- Any assignment of truth values ϕ corresponds naturally to a subset of $\{t_1, f_1, \ldots, t_m, f_m\}$, namely the set
$$S_\phi := \{t_j \mid \phi(p_j) = \texttt{true}\} \cup \{f_j \mid \phi(p_j) = \texttt{false}\}. \tag{4.2.1}$$
- If ϕ makes D_i true then S_ϕ is a subset of A_i.
- If S_ϕ is a subset of some set A_i then S_ϕ is closed.

To prove that FIRST-N-PI is at least as hard as VALIDITY, we need to show two things. First, we need to show that $\mathbb{K}_f$ and $P_1, \ldots, P_m$ indeed constitute an instance of FIRST-N-PI and second, we need to show that f is valid if and only if $P_1, \ldots, P_m$ are lectically the first m pseudo-intents of $\mathbb{K}_f$.

Lemma 20. *$\mathbb{K}_f$ and $P_1, \ldots, P_m$ constitute an instance of* FIRST-N-PI.

Proof. We have to show that $P_j, j \in \{1, \ldots, m\}$, are pseudo-intents of $\mathbb{K}_f$. Note that all strict subsets of P_j are closed in $\mathbb{K}_f$ (this is because all singleton subsets $\{t_j\}$ and $\{f_j\}$ are object intents of some u_r). To see that $\alpha_j \in P_j''$ and thus $P_j'' \neq P_j$ consider the sets A_i for $i \in \{1, \ldots, k\}$. If $P_j = \{t_j, f_j\} \subseteq A_i$ then by definition of A_i, p_j does not occur in D_i. Therefore $\alpha_j \in A_i$. Let $s \in \{1, \ldots, m\}$ be an index of some set T_{is}. If $P_j \subseteq T_{is}$ then $P_j \subseteq A_i$ and $j \neq s$. Then $\alpha_j \in A_i$ holds and because $j \neq s$ it follows that $\alpha_j \in T_{is} = A_i \setminus \{f_s, \alpha_s\}$. Analogously $\alpha_j \in F_{is}$ if $P_j \subseteq F_{is}$. Therefore all objects that have all attributes from P_j also have α_j as an attribute, and thus $\alpha_j \in P_j''$ holds. Therefore $P_j'' \neq P_j$ holds. Hence P_j is a pseudo-intent. Hence $\mathbb{K}_f$ and $P_1, \ldots, P_m$ indeed constitute an instance of FIRST-N-PI. $\square$

Next we show that $\mathbb{K}_f$ has a pseudo-intent that is lectically smaller than P_1 if and only if f is not valid. For this we first show some auxiliary lemmas. Let ϕ be an assignment that maps all variables p_j to a truth value in $\{\texttt{true}, \texttt{false}\}$. Let S_ϕ be defined as in Eq. (4.2.1). Note that S_ϕ contains exactly one element of $\{t_j, f_j\}$ for every $j \in \{1, \ldots, m\}$.

Lemma 21. *There is some $i \in \{1, \ldots, k\}$ for which $S_\phi \subseteq A_i$ if and only if $f(\phi(p_1), \ldots, \phi(p_m)) = \texttt{true}$.*

Proof. ($\Rightarrow$) Let ϕ be such that $S_\phi \subseteq A_i$. Then by definition of A_i it holds that $f_j \notin S_\phi$, and thus $\phi(p_j) = \texttt{true}$, for all p_j that occur as positive literals in D_i (we have removed f_j from A_i). Analogously, $\phi(p_j) = \texttt{false}$ for all p_j that occur as negative literals. Hence all literals in D_i evaluate to $\texttt{true}$ and therefore both D_i and the whole formula evaluate to $\texttt{true}$.

($\Leftarrow$) Now let ϕ be an assignment that makes f true. Since f is in DNF it evaluates to $\texttt{true}$ iff at least one of the k implicants evaluates to true. Let D_i for some $i \in \{1, \ldots, k\}$ be an implicant that evaluates to true. Then $\phi(p_j) = \texttt{true}$ for all p_j that occur as positive literals in D_i and $\phi(p_j) = \texttt{false}$ for all p_j that occur as negative literals in D_i. By definition of A_i and S_ϕ this implies $S_\phi \subseteq A_i$. $\square$

Lemma 22. *If $S_\phi \subseteq A_i$ then S_ϕ can be written as*

$$S_\phi = \bigcap_{\substack{j \in \{1,\ldots,m\} \\ \phi(p_j)=\texttt{true}}} T_{ij} \cap \bigcap_{\substack{j \in \{1,\ldots,m\} \\ \phi(p_j)=\texttt{false}}} F_{ij}.$$

Proof. We denote the right hand side of the above equation by R. By definition S_ϕ does not contain f_j if $\phi(p_j) = \texttt{true}$. Thus $S_\phi \subseteq A_i - \{f_j, \alpha_j\} = T_{ij}$ for all $j \in \{1, \ldots, m\}$ for which $\phi(p_j) = \texttt{true}$. Likewise, $S_\phi \subseteq A_i - \{t_j, \alpha_j\} = F_{ij}$ for all $j \in \{1, \ldots, m\}$ for which $\phi(p_j) = \texttt{false}$. Thus $S_\phi \subseteq R$. To prove the other inclusion consider some $x \in R$. For every $j \in \{1, \ldots, m\}$ it holds that $\alpha_j \notin F_{ij}$ and $\alpha_j \notin T_{ij}$. If $\phi(p_j) = \texttt{true}$ then $R \subseteq T_{ij}$, otherwise $R \subseteq F_{ij}$. So in either case $\alpha_j \notin R$. Therefore $x \neq \alpha_j$ holds for all $j \in \{1, \ldots, m\}$. Assume that $x = t_j$ for some j. Then $\phi(p_j) = \texttt{true}$ must hold, for otherwise R would be a subset of F_{ij} which does not contain t_j. Now $\phi(p_j) = \texttt{true}$ implies $x = t_j \in S_\phi$. The case $x = f_j$ for some j can be treated analogously. Thus for every $x \in R$ it holds that $x \in S_\phi$ and thus $R \subseteq S_\phi$. Hence $S_\phi = R$. $\square$

Lemma 23. *f is valid if and only if for all assignments ϕ the set S_ϕ is closed in $\mathbb{K}_f$.*

Proof. ($\Leftarrow$) Assume that there is an assignment ϕ that makes f false. From Lemma 21 it follows that $S_\phi \not\subseteq A_i$ for all $i \in \{1, \dots, k\}$. But then no object in G has all the attributes in S_ϕ because every object intent is either a singleton set or a subset of some A_i. Therefore $S''_\phi = M$ and thus S_ϕ is not closed. This contradicts the assumption, and thus f must be valid.

($\Rightarrow$) Assume that there is some ϕ for which S_ϕ is not closed. We know that the intersection of closed sets is also closed. This implies in particular that S_ϕ cannot be written as the intersection of object intents. From Lemma 22 it follows that $S_\phi \not\subseteq A_i$ for all $i \in \{1, \dots, k\}$. But then Lemma 21 shows that ϕ makes f false. This is a contradiction to the assumption that f is valid. Therefore S_ϕ must be closed for all ϕ. □

Lemma 24. *$P_1, \dots, P_m$ are lectically the smallest pseudo-intents of $\mathbb{K}_f$ if and only if for all assignments ϕ the set S_ϕ is closed in $\mathbb{K}_f$.*

Proof. ($\Rightarrow$) Assume that some S_ϕ is not closed. By Proposition 11 there is a pseudo-intent $P \subseteq S_\phi$. The definition of S_ϕ shows that $\alpha_j \notin P$ and $\{t_j, f_j\} \not\subseteq P$ for all $j \in \{1, \dots, m\}$. Therefore, the smallest attribute that distinguishes P and P_1 must be either t_1 or f_1; both are in P_1. Thus, P is lectically smaller than P_1. Also P must be different from all the P_j because S_ϕ does not include any of the P_j. This is a contradiction to the assumption that $P_1, \dots, P_m$ are the lectically smallest pseudo-intents of $\mathbb{K}$.

($\Leftarrow$) Let $Q \subseteq M$ be a set of attributes that is lectically smaller than P_1. Then Q cannot contain any α_j because otherwise Q would be lectically larger than P_1. Therefore Q must be a subset of $\{t_1, f_1, \dots, t_m, f_m\}$. *Case* 1: *There is some $j \in \{1, \dots, m\}$ such that $P_j \subseteq Q$*. Then $\alpha_j \in P''_j \setminus Q$ and thus $P''_j \not\subseteq Q$. Therefore either Q is equal to P_j or Q is not a pseudo-intent. *Case* 2: *For all $j \in \{1, \dots, m\}$ it holds that $P_j \not\subseteq Q$*. Define

$$\phi_t(p_j) = \begin{cases} \texttt{true} & t_j \in Q \\ \texttt{false} & f_j \in Q \\ \texttt{true} & \text{otherwise} \end{cases} \qquad \phi_f(p_j) = \begin{cases} \texttt{true} & t_j \in Q \\ \texttt{false} & f_j \in Q \\ \texttt{false} & \text{otherwise.} \end{cases}$$

Both ϕ_t and ϕ_f are well-defined since Q cannot contain both t_j and f_j for any j. With ϕ_t and ϕ_f defined as above it holds that $Q = S_{\phi_t} \cap S_{\phi_f}$. Since all S_ϕ are closed the intersection of S_{ϕ_t} and S_{ϕ_f} must also be closed. Therefore Q cannot be a pseudo-intent. □

Theorem 25 (*Hardness of* FIRST-N-PI)**.** FIRST-N-PI *is* CONP*-hard.*

Proof. From Lemmas 23 and 24 it follows that $P_1, \dots, P_m$ are lectically the first pseudo-intents of $\mathbb{K}_f$ if and only if f is valid. Since the reduction we have given can be done in polynomial time, and VALIDITY is CONP-hard, it follows that FIRST-N-PI is CONP-hard. □

The following is an immediate consequence of Lemma 19 and Theorem 25.

Corollary 26. FIRST-N-PI *is* CONP*-complete.*

What consequence does Theorem 25 have for enumerating pseudo-intents in a given lectic order? In order to understand this, assume that there is an algorithm $\mathcal{A}$ that given a context enumerates its pseudo-intents in the lectic order and with polynomial delay. This means that there is a polynomial $p(|G|, |M|)$ such that the delay between the computation of two consequent pseudo-intents is bounded by $p(|G|, |M|)$.

In order to solve FIRST-N-PI for an input context $\mathbb{K} = (G, M, I)$ and the sets $P_1, \dots, P_n$, one can construct a new algorithm $\mathcal{A}'$ that works as follows: $\mathcal{A}'$ lets $\mathcal{A}$ run for time $n \cdot p(|G|, |M|)$. Upon termination, $\mathcal{A}$ outputs lectically the first n pseudo-intents of $\mathbb{K}$. $\mathcal{A}'$ compares the output of $\mathcal{A}$ with $P_1, \dots, P_n$ and returns *yes* if they are equal, and *no* otherwise. The runtime of $\mathcal{A}'$ is bounded by $n \cdot p(|G|, |M|)$, i.e., polynomial in the size of the input. This means that if we had an algorithm that enumerates the pseudo-intents of a given context in the lectic order with polynomial delay, then using this algorithm we could solve the CONP-hard problem FIRST-N-PI in polynomial time.

Theorem 27. *Unless* P = NP, *pseudo-intents cannot be enumerated in the lectic order with polynomial delay.*

It turns out that unlike concept intents, pseudo-intents cannot be efficiently enumerated in the lectic order. Having this result, the next question if of course what happens if we remove the restriction on the order of the output. Does the problem become easier if we do not require them to be output in the lectic order? We try to answer these questions next.

5. Complexity of enumerating pseudo-intents without order

In the present section we investigate whether pseudo-intents can be enumerated in output-polynomial time. Our problem is defined as:

Problem: PSEUDO-INTENT ENUMERATION (PIE)

Input: A formal context $\mathbb{K}$.

Output: The set of pseudo-intents of $\mathbb{K}$.

In order to investigate its complexity, we formalize two decision problems, and analyze their complexities in the next sections.

5.1. *Checking the existence of an additional pseudo-intent*

Our first decision problem is to decide for a given context and a given set of pseudo-intents of it, whether there is an additional pseudo-intent. As we will see, it has crucial importance for determining the complexity of PIE.

Problem: ADDITIONAL PSEUDO-INTENT (API)
Input: A formal context $\mathbb{K} = (G, M, I)$, and a set $\mathcal{P}$ of pseudo-intents of $\mathbb{K}$, i.e., $\mathcal{P} \subseteq \{P \mid P \subseteq M, P \text{ pseudo-intent of } \mathbb{K}\}$.
Question: Is there an additional pseudo-intent, i.e., $Q \subseteq M$ s.t. Q is a pseudo-intent of $\mathbb{K}$ and $Q \notin \mathcal{P}$?

Proposition 28 says that if this problem cannot be decided in polynomial time then, unless P = NP, PIE cannot be solved in output polynomial time. Its proof is based on a generic argument that for instance can be found in [28,38]. Therefore we are not going to give its proof here.

Proposition 28. *If* API *cannot be decided in polynomial time, then unless* P = NP, PIE *cannot be solved in output-polynomial time.*

The proposition shows that determining the complexity of API is indeed crucial for determining the complexity of PIE. We can decide API by using the algorithm in [30]. We first non-deterministically guess a $Q \subseteq M$, then using this algorithm check whether Q is a pseudo-intent. However, this gives us an NP algorithm with a CONP oracle, i.e., its complexity would be NP^{CONP}, since the algorithm in [30] is a CONP algorithm. As we will see next, it is actually possible to decide API in NP.

Proposition 29. API *is in* NP.

Proof. Given an instance of API with the input $\mathbb{K}$ and $\mathcal{P}$, construct the set of implications $\mathcal{L} = \{P \to P'' \mid P \in \mathcal{P}\}$ and non-deterministically guess a set $Q \subseteq M$. We can verify in polynomial time that $Q \to Q''$ *does not* follow from $\mathcal{L}$, i.e., there is a pseudo-intent that is not in $\mathcal{P}$. □

In the following we investigate the lower complexity bound for this problem. We show that it is at least as hard as the complement of a prominent open problem on hypergraphs. However, whether API is NP-hard remains unfortunately open.

We first need to introduce some more notions from hypergraphs. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called *saturated* [8] if every subset of V is contained in at least one of the edges of $\mathcal{H}$, or it contains at least one edge of $\mathcal{H}$, i.e., for every $W \subseteq V$, $W \subseteq E$ holds, or $E \subseteq W$ holds for some $E \in \mathcal{E}$. It has been shown in [8] that checking whether a hypergraph is saturated is CONP-complete. There, a special case of the problem, where the given hypergraph is restricted to be simple, has also been considered. A hypergraph is called *simple* if no edge contains another edge.

Problem: SIMPLE HYPERGRAPH SATURATION (SIMPLE-H-SAT)
Input: A simple hypergraph $\mathcal{H} = (V, \mathcal{E})$
Question: Is $\mathcal{H}$ saturated?

It is not difficult to see that this problem is in CONP. However, up to now there has neither been a proof that it is CONP-hard, nor a proof that it is in P. It has been shown in [8] that this problem is under polynomial transformations computationally equivalent to TRANS-HYP. In the following we show that API is at least as hard as the complement of SIMPLE-H-SAT:

Theorem 30. API *is co*SIMPLE-H-SAT*-hard.*

Proof. Let an instance of SIMPLE-H-SAT be given by the simple hypergraph $\mathcal{H} = (V, \mathcal{E})$, where $\mathcal{E} = \{E_1, \ldots, E_n\}$. From $\mathcal{H}$ we construct the context $\mathbb{K}_{\mathcal{H}} = (G, M, I)$, where $M = V$, and G and I are defined as follows: For every E_i, $1 \le i \le n$, we create the following objects: For every $D \subsetneq E_i$ such that $|D| = |E_i| - 1$, we create an object with the intent D. E_i has $|E_i|$-many such subsets. We denote these objects by g_{ij}, where $1 \le i \le n$ and $1 \le j \le |E_i|$. In total, G contains $\sum_{i=1}^{n} |E_i|$ objects. We construct $\mathcal{P}$ by just taking the edges of $\mathcal{H}$, i.e., $\mathcal{P} = \{E_1, \ldots, E_n\}$. Obviously, both $\mathbb{K}_{\mathcal{H}}$ and $\mathcal{P}$ can be constructed in time polynomial in the size of $\mathcal{H}$.

Note that $\mathbb{K}_{\mathcal{H}}$ has the following property: Since $\mathcal{H}$ is simple, no edge is contained in another edge, and obviously not in a strict subset of another edge. Then, for every $i \in \{1, \ldots, n\}$, $E_i' = \emptyset$ and $E_i'' = M$ holds. That is E_i is not closed. Moreover every strict subset $D \subsetneq E_i$ can be written as the intersection of concept intents of the form $\{g_{ij}\}'$. Hence all strict subsets of an edge E_i are closed. Thus, the edges E_i are pseudo-intents of $\mathbb{K}_{\mathcal{H}}$, which means that $\mathbb{K}_{\mathcal{H}}$ and $\mathcal{P}$ indeed form an instance of API. We claim that $\mathcal{H}$ is *not* saturated if and only if $\mathbb{K}_{\mathcal{H}}$ has an additional pseudo-intent.

($\Rightarrow$) Assume $\mathcal{H}$ is not saturated. Then, there exists a $W \subseteq V$ such that for every $i \in \{1, \ldots, n\}$, $W \not\subseteq E_i$ holds and $E_i \not\subseteq W$ holds. Assume without loss of generality that W is minimal with respect to the property $W \not\subseteq E_i$ for every $1 \le i \le n$. Since W is not contained in any E_i, and obviously not contained in any strict subset of any E_i, $W' = \emptyset$ and $W'' = M$. That is W is not closed. Take any $X \subsetneq W$. Since W is minimal, $X \subseteq E_i$ holds for some $1 \le i \le n$. We know that $E_i \not\subseteq W$, then $X = E_i$ cannot hold, thus X satisfies $X \subsetneq E_i$. Since all strict subsets of E_i are closed, X is closed. We have shown that W is not closed but all its strict subsets are closed, thus W is a pseudo-intent. Moreover, it is an additional pseudo-intent since $W \neq E_j$, for all $1 \le j \le n$.

($\Leftarrow$) Assume $\mathbb{K}_{\mathcal{H}}$ has an additional pseudo-intent, i.e., a pseudo-intent Q such that $Q \neq E_i$ for every $1 \le i \le n$. Since strict subsets of E_i are closed, Q cannot be a strict subset of any E_i. Thus $Q \not\subseteq E_i$ for every $1 \le i \le n$. Moreover, by definition Q contains the closure of strictly smaller pseudo-intents. We know that for every $1 \le i \le n$, E_i is a pseudo-intent, and

$E_i'' = M$. Since Q does not strictly contain M, it cannot strictly contain any E_i either. Together with $Q \neq E_i$, this implies that $E_i \not\subseteq Q$. We have shown that there exists a $Q \subseteq V$ such that $Q \not\subseteq E_i$ and $E_i \not\subseteq Q$ for every $1 \leq i \leq n$, thus $\mathcal{H}$ is not saturated. □

The following is an immediate consequence of Theorem 30 above and Theorem 4.12 in [8]:

Corollary 31. API *is co*TRANS-HYP*-hard.*

5.2. Recognizing the set of pseudo-intents

The second decision problem we consider for analyzing PIE is deciding whether a given set of pseudo-intents is precisely the set of all pseudo-intents of a given context.

Problem: PSEUDO-INTENTS (PIS)
Input: A formal context $\mathbb{K} = (G, M, I)$, and a set $\mathcal{P} \subseteq \mathscr{P}M$.
Question: Is $\mathcal{P}$ precisely the set of pseudo-intents of $\mathbb{K}$?
As in the case of API, determining the complexity of PIS is important for determining the complexity of PIE.

Proposition 32. *If* PIS *cannot be decided in polynomial time, then unless* P = NP, PIE *cannot be solved in output-polynomial time.*

The proof of Proposition 32 is based on the same generic argument as the proof of Proposition 28. Therefore we here we leave out the proof, which can be found in [38].

Proposition 33. PIS *is in co*NP.

Proof. Given an instance with the input $\mathbb{K} = (G, M, I)$ and $\mathcal{P}$, an algorithm that decides PIS for this instance first checks whether the elements of $\mathcal{P}$ are pseudo-intents of $\mathbb{K}$. If it encounters an element that is not a pseudo-intent, it terminates and returns *no*. If every $P \in \mathcal{P}$ is a pseudo-intent, then it continues as in the algorithm in the proof of Proposition 29. The algorithm constructs the set of implications $\mathcal{L} = \{P \to P'' | P \in \mathcal{P}\}$ and non-deterministically guesses a set $Q \subseteq M$. Obviously the implication $Q \to Q''$ holds in $\mathbb{K}$, thus if $\mathcal{L}$ is a base for $\mathbb{K}$ then $Q \to Q''$ follows from $\mathcal{L}$. Then the algorithm verifies that this is *not* the case.

It is not difficult to see that this is a coNP algorithm. In the first step the algorithm performs polynomially-many checks, each of which can be done in coNP by using the algorithm in [30]. In the second step the algorithm non-deterministically guesses a Q and in polynomial time verifies that $Q \to Q''$ does *not* follow from $\mathcal{L}$, which means that $\mathcal{L}$ is *not* a base, which implies that $\mathcal{P}$ is *not* the set of all pseudo-intents of $\mathbb{K}$. This step can be performed in coNP as well, thus the whole algorithm is a coNP algorithm. □

For the lower bound of PIS, we just use our result on the lower bound of API. In fact it is not difficult to see that PIS is at least as hard as the complement of API. We take an instance of API and create an instance of PIS just by taking the same context and the set of pseudo-intents as the set of subsets of M. This set is precisely the set of pseudo-intents if and only if there is *no* additional pseudo-intent. Thus PIS is coAPI-hard. Since we have shown in Corollary 31 that API is coTRANS-HYP-hard, it follows that PIS is TRANS-HYP-hard.

Corollary 34. PIS *is* TRANS-HYP*-hard.*

6. Recognizing and enumerating minimal pseudo-intents

In the present section we restrict our attention to the pseudo-intents that we call minimal pseudo-intents, and analyze the complexity of recognizing and enumerating them. We say that P is a *minimal pseudo-intent of* $\mathbb{K}$ if P is a pseudo-intent of $\mathbb{K}$ and P does not contain any other pseudo-intent of $\mathbb{K}$. An equivalent definition is the following.

Definition 35 (*Minimal Pseudo-Intent*)**.** A *minimal pseudo-intent* of a context is a set $P \subseteq M$ such that

- P is not closed, and
- every strict subset $S \subset P$ is closed.

Minimal pseudo-intents play an important role in implicational bases. They occur as premises not only in the Duquenne–Guigues Base, but also in all other bases of a formal context. In order to see this assume that $\mathcal{L}$ is an implicational base for the context $\mathbb{K}$, P is a minimal pseudo-intent of $\mathbb{K}$, and $\mathcal{L}$ contains no implication with the premise P. Since all strict subsets of P are closed, there can be no implication $C \to D$ in $\mathcal{L}$ such that $C \subseteq P$ and $D \not\subseteq P$. But then $P \to P''$ does not follow from $\mathcal{L}$, and thus $\mathcal{L}$ is not complete, which contradicts our initial assumption that $\mathcal{L}$ is an implicational base for $\mathbb{K}$.

Lemma 36. *If $\mathcal{L}$ is an implication base of a given context $\mathbb{K} = (G, M, I)$ and P is a minimal pseudo-intent of $\mathbb{K}$, then $\mathcal{L}$ contains an implication $P \to D$ where $D \subseteq M$.*

6.1. *Recognizing minimal pseudo-intents*

As we have already mentioned in Section 3, the best known algorithm for recognizing pseudo-intents runs in coNP[30,31]. In the following we show that for minimal pseudo-intents the problem is solvable in polynomial time. Our algorithm is based on the following simple property.

Lemma 37. *All strict subsets of a set P are closed if and only if all sets $P \setminus \{m\}$, where $m \in P$, are closed.*

Proof. Assume that all sets of the form $P \setminus \{m\}$, where $m \in P$, are closed. Take a proper subset $S \subsetneq P$. S can be written as the intersection $S = \bigcap_{m \in P \setminus S}(P \setminus \{m\})$. Since the intersection of closed sets is itself closed, S must be closed. The other direction of the claim is trivial. □

Lemma 37 enables us to recognize a minimal pseudo-intent without testing all of its proper subsets for closedness. In order to check whether a given set P is a minimal pseudo-intent it suffices to check whether the subsets $P \setminus \{m\}$, where $m \in P$, are closed, and P itself is not closed. Since this requires only polynomially many closedness tests and each such test is polynomial, the whole test can be done in polynomial time.

6.2. *Enumerating minimal pseudo-intents without order*

In Section 5 we have shown that enumerating pseudo-intents of a context without order is at least as hard as enumerating minimal transversals of a hypergraph. In the present section we investigate the complexity of this enumeration problem for minimal pseudo-intents. Interestingly, it turns out that this problem is not solvable in output polynomial time unless P = NP.

Like in Section 5, we define a decision problem associated to the enumeration problem we are going to investigate.
Problem: All minimal pseudo-intents (all-mpi)

Input: A formal context $\mathbb{K} = (G, M, I)$ and a set of minimal pseudo-intents $\mathcal{P}$.
Question: Is $\mathcal{P}$ the set of all minimal pseudo-intents of $\mathbb{K}$?
It turns out that this problem is coNP-complete.

Theorem 38. all-mpi *is coNP-complete.*

Proof. We prove that the problem is in coNP. By Lemma 37 we know that testing whether a set $Q \subseteq M$ is a minimal pseudo-intent can be done in polynomial time. In order to decide whether $\mathcal{P}$ contains all minimal pseudo-intents of $\mathbb{K}$, one can non-deterministically guess a set $Q \subseteq M$ such that $Q \notin \mathcal{P}$ and then check in polynomial time whether it is a minimal pseudo-intent. Thus the dual problem of all-mpi can be decided in non-deterministic polynomial time. Therefore all-mpi is in coNP.

For showing hardness we use the same construction as in the proof of Theorem 25. Given an instance of validity with the propositional formula f in DNF, we construct the same context $\mathbb{K}_f$ shown in Table 3. Additionally, we construct $\mathcal{P} = \{P_1, \ldots, \mathcal{P}_{2m}\}$ by defining $P_1 = \{t_1, f_1\}, \ldots, P_m = \{t_m, f_m\}$, $P_{m+1} = \{\alpha_1\}, \ldots, P_{2m} = \{\alpha_m\}$. In the proof of Theorem 25 we have already shown that $P_1, \ldots, P_m$ are minimal pseudo-intents. Let $g \in G$ be an object and let $j \in \{1, \ldots, m\}$. By construction of $\mathbb{K}_f$ the attribute α_j is contained in g' if and only if $\{t_j, f_j\} \subseteq g'$. Therefore $\{t_j, f_j\}$ is contained in $\{\alpha_j\}''$, i.e., $\{\alpha_j\}$ is not closed. Since the only proper subset of a set $\{\alpha_j\}$, namely the empty set, is closed in $\mathbb{K}_f$, $P_{m+1} = \{\alpha_1\}, \ldots, P_{2m} = \{\alpha_m\}$ are minimal pseudo-intents. $\mathbb{K}_f$ and $\mathcal{P}$ constructed this way indeed form an instance of all-mpi. We claim that $\mathcal{P}$ contains all minimal pseudo-intents of $\mathbb{K}_f$ if and only if f is valid. Using Lemma 23 we can reformulate this claim to the following: $P_1, \ldots, P_{2m}$ are all minimal pseudo-intents of $\mathbb{K}$ iff for all assignments ϕ the set S_ϕ is closed in $\mathbb{K}$.

($\Rightarrow$) Assume that some S_ϕ is not closed. Then S_ϕ must contain a minimal pseudo-intent $P \subseteq S_\phi$. The definition of S_ϕ (Eq. (4.2.1)) shows that S_ϕ does not contain α_j, and it contains either t_j or f_j but not both, for all $j \in \{1, \ldots, m\}$. Thus S_ϕ does not contain any of the $P_1, \ldots, P_{2m}$. Therefore the minimal pseudo-intent P must be an additional minimal pseudo-intent. This contradicts the assumption that S_ϕ is not closed, and therefore S_ϕ must be closed.

($\Leftarrow$) Assume that there is an additional minimal pseudo-intent $Q \subseteq M$, i.e. Q is a minimal pseudo-intent such that $Q \neq P_i$ for all $j \in \{1, \ldots, 2m\}$. If Q contains some α_j then Q cannot be a minimal pseudo-intent. Therefore Q is a subset of $\{t_1, f_1, \ldots, t_m, f_m\}$. If Q contains $\{t_1, f_1\}$ then Q cannot be a minimal pseudo-intent either. Hence the smallest attribute that distinguishes Q and $P_1 = \{t_1, f_1\}$ is contained in P_1, i.e. Q is lectically smaller than P_1. This contradicts Lemma 24, which states that such a pseudo-intent Q cannot exist if all sets S_ϕ are closed.

We have now shown that $P_1, \ldots, P_{2m}$ are all minimal pseudo-intents of $\mathbb{K}_f$ iff for all assignments ϕ the set S_ϕ is closed in $\mathbb{K}_f$. Together with Lemma 23 this implies that $P_1, \ldots, P_{2m}$ are all minimal pseudo-intents of $\mathbb{K}_f$ iff f is valid, and thus all-mpi is coNP-hard. □

Theorem 38 implies that our original enumeration problem, namely enumerating minimal pseudo-intents, cannot be solved in output polynomial time. The proof of this claim is based on the general argument used also in the proof of Proposition 28.

Corollary 39. *The set of all minimal pseudo-intents of a context $\mathbb{K}$ cannot be computed in output-polynomial time unless* P = NP.

Table 4
Context $\mathbb{K}_f^*$.

	x	y	α_1	$\cdots$	α_m	t_1	f_1	$\cdots$	t_m	f_m
u_1						X				
$\vdots$								$\cdots$		
u_{2m}										X
$g_{T_{11}}$	X				$\cdots$	T_{11}	$\cdots$			
$\vdots$	$\vdots$					$\vdots$				
$g_{T_{km}}$	X				$\cdots$	T_{km}	$\cdots$			
$g_{F_{11}}$	X				$\cdots$	F_{11}	$\cdots$			
$\vdots$	$\vdots$					$\vdots$				
$g_{F_{km}}$	X				$\cdots$	F_{km}	$\cdots$			
$g_{T_{01}}$					$\cdots$	T_{01}	$\cdots$			
$\vdots$						$\vdots$				
$g_{T_{0m}}$					$\cdots$	T_{0m}	$\cdots$			
$g_{F_{01}}$					$\cdots$	F_{01}	$\cdots$			
$\vdots$						$\vdots$				
$g_{F_{0m}}$					$\cdots$	F_{0m}	$\cdots$			

At first glance one might think that this result also holds for enumerating all pseudo-intents, and not only the minimal ones. However, a closer look reveals that this is unfortunately not the case. It would be the case if the total number of pseudo-intents of a context were bounded by a polynomial in the number of its minimal pseudo-intents. Currently we do not know whether this holds but we conjecture it does not. It is definitely an interesting point to investigate in the future.

6.3. *Computing the lectically last minimal pseudo-intent*

We have seen that it is possible to find the lectically first pseudo-intent in polynomial time. Since the lectic order extends the subset order, the lectically first pseudo-intent of a context is also the lectically first minimal pseudo-intent of a context. A related problem is finding the lectically *last* pseudo-intent. We cannot say anything about the complexity of this problem for pseudo-intents in general. However, we can modify the context from Table 3 to yield a reduction for the corresponding problem for minimal pseudo-intents, or more precisely the decision version of it:

Problem: LECTICALLY LAST MINIMAL PSEUDO-INTENT (LAST-MPI)

Input: A formal context $\mathbb{K} = (G, M, I)$ and a pseudo-intent P.

Question: Is P the lectically last minimal pseudo-intent of $\mathbb{K}$?

It is not hard to see that LAST-MPI is in CONP. One can non-deterministically guess a set of attributes $A \subseteq M$ and check in polynomial time whether A is lectically larger than P, and by using Lemma 37 whether it is a minimal pseudo-intent. We shall see that LAST-MPI is closely related to the following decision problem.

Problem: ELEMENT OF A MINIMAL PSEUDO-INTENT (ϵ-MPI)

Input: A formal context $\mathbb{K} = (G, M, I)$ and an attribute $a \in M$.

Question: Is there a minimal pseudo-intent P of $\mathbb{K}$ such that $a \in P$?

ϵ-MPI can be decided in non-deterministic polynomial time. One can non-deterministically guess a set of attributes $A \subseteq M$ and check whether $a \in A$ holds and A is a minimal pseudo-intent.

Lemma 40. LAST-MPI *is in* CONP *and* ϵ-MPI *is in* NP.

In order to show hardness for these two related problems, we again use a reduction from VALIDITY. Let f be a formula $f(p_1, \ldots, p_m) = D_1 \vee \cdots \vee D_k$, where $D_i = (x_{i1} \wedge \cdots \wedge x_{il_i})$ and $x_{ir} \in \{p_1, \ldots, p_m\} \cup \{\neg p_1, \ldots, \neg p_m\}$ for all $i \in \{1, \ldots, k\}$ and all $r \in \{1, \ldots, l_i\}$. We construct a context $\mathbb{K}_f^* = (G, M^*, I^*)$ (Table 4) which is obtained from $\mathbb{K}_f = (G, M, I)$ (Table 3) by adding attributes x and y to M and adding objects $g_{T_{0j}}$ and $g_{F_{0j}}$ for all $j \in \{1, \ldots, m\}$. The incidence relation I^* of $\mathbb{K}_f^*$ is obtained from I by adding the pairs $(x, g_{T_{ij}})$ and $(x, g_{F_{ij}})$ for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, m\}$. For all $j \in \{1, \ldots, m\}$ the object intents of the new objects $g_{T_{0j}}$ and $g_{F_{0j}}$ are defined to be $g_{T_{0j}}^{I^*} = T_{0j}$ and $g_{T_{0j}}^{I^*} = F_{0j}$, respectively, where

$$T_{0j} = M \setminus \{x, y\} \setminus \{\alpha_j, f_j\},$$
$$F_{0j} = M \setminus \{x, y\} \setminus \{\alpha_j, t_j\}.$$

Proposition 41. *Let ϕ be any assignment of truth values to the propositional variables $p_1, \ldots, p_m$, and S_ϕ be defined as in Eq. (4.2.1). Then all subsets $R \subseteq S_\phi$ are closed in $\mathbb{K}_f^*$. Furthermore, for all $i \in \{1, \ldots, k\}$ and for every set $Q \subseteq S_\phi \cap A_i$ the set $\{x\} \cup Q$ is closed in $\mathbb{K}_f^*$.*

Proof. R can be written as

$$R = \bigcap_{\substack{j\in\{1,\ldots,m\}\\ f_j \notin R}} T_{0j} \cap \bigcap_{\substack{j\in\{1,\ldots,m\}\\ t_j \notin R}} F_{0j} = \bigcap_{\substack{j\in\{1,\ldots,m\}\\ f_j \notin R}} g'_{T_{0j}} \cap \bigcap_{\substack{j\in\{1,\ldots,m\}\\ t_j \notin R}} g'_{F_{0j}},$$

while Q can be written as

$$Q = \bigcap_{\substack{j\in\{1,\ldots,m\}\\ f_j \notin Q}} T_{ij} \cap \bigcap_{\substack{j\in\{1,\ldots,m\}\\ t_j \notin Q}} F_{ij}$$

and thus

$$\begin{aligned}\{x\} \cup Q &= \bigcap_{\substack{j\in\{1,\ldots,m\}\\ f_j \notin Q}} \{x\} \cup T_{ij} \cap \bigcap_{\substack{j\in\{1,\ldots,m\}\\ t_j \notin Q}} \{x\} \cup F_{ij}\\ &= \bigcap_{\substack{j\in\{1,\ldots,m\}\\ f_j \notin Q}} g'_{T_{ij}} \cap \bigcap_{\substack{j\in\{1,\ldots,m\}\\ t_j \notin Q}} g'_{F_{ij}}.\end{aligned}$$

Therefore both R and $\{x\} \cup Q$ can be written as the intersection of concept intents and therefore they are closed. □

Lemma 42. *$\mathbb{K}_f^*$ has a minimal pseudo-intent that contains x iff f is not valid.*

Proof. ($\Leftarrow$) If f is not valid then there is a truth assignment ϕ that makes f false. Then by Lemma 21, $S_\phi \not\subseteq A_i$ holds for all $i \in \{1, \ldots, k\}$. Let Q be a minimal subset of S_ϕ such that $Q \not\subseteq A_i$ still holds for all $i \in \{1, \ldots, k\}$. Define $P = \{x\} \cup Q$. P is not closed (it is not contained in any concept intent), but every strict subset of P is closed because of Proposition 41. Therefore P is a minimal pseudo-intent that contains x.

($\Rightarrow$) Using the same arguments as in the proofs of Lemma 20 and Theorem 38 we can show that $\{t_1, f_1\}, \ldots, \{t_m, f_m\}$ and $\{\alpha_1\}, \ldots, \{\alpha_m\}$ are minimal pseudo-intents. Furthermore, it holds that $\{y\}'' = M^*$ and $\emptyset'' = \emptyset$. Therefore $\{y\}$ is a minimal pseudo-intent, too. If $\mathbb{K}_f^*$ has a minimal pseudo-intent P that contains x then by definition P does not contain another pseudo-intent. Thus $P = \{x\} \cup Q$, where $Q \subseteq S_\phi$ for some assignment ϕ. Since P is a minimal pseudo-intent it cannot be closed. By Proposition 41 this implies that Q is not a subset of A_i for $1 \le i \le k$. Thus for all $1 \le i \le k$ it holds that $S \not\subseteq A_i$. It follows from Lemma 21 that S_ϕ makes f false, i.e., f is not valid. □

We need the following proposition for completing the proof of hardness of LAST-MPI.

Corollary 43. *Let the order on the attributes be $x < y < \alpha_1 < \cdots < \alpha_m < t_1 < f_1 < \cdots < t_m < f_m$. $\{y\}$ is the lectically greatest minimal pseudo-intent of $\mathbb{K}_f^*$ iff f is valid.*

Proof. $\{y\}$ is the lectically greatest minimal pseudo-intent iff there is no minimal pseudo-intent that contains $\{x\}$. The rest follows from Lemma 42. □

Theorem 44. LAST-MPI *is* CONP*-complete and* ϵ-MPI *is* NP*-complete.*

Proof. Hardness of ϵ-MPI follows from Lemma 42, and hardness of LAST-MPI follows from Lemma 42 and Corollary 43. Together with Lemma 40 this shows completeness for ϵ-MPI and LAST-MPI. □

7. Concluding remarks and future work

We have investigated the computational complexity of the problem of enumerating the pseudo-intents of a given formal context, which we have called PIE. We have first considered enumeration in a specified lexicographic order, and shown that lexicographically the first pseudo-intent can be computed in polynomial time (Algorithm 1). In order to investigate whether the remaining ones can efficiently be enumerated, we have formulated a decision problem, namely FIRST-N-PI, which is the problem of checking whether n given pseudo-intents are lexicographically the first n ones. It turned out that this problem is CONP-complete (Corollary 26), which implies that unless P = NP all pseudo-intents of a context cannot be enumerated in lexicographic order with polynomial delay (Theorem 27).

Later we have removed the restriction on the order of output, and investigated whether in this setting the problem can be solved in output polynomial time. To this purpose we have formulated two decision problems, namely API and PIS, whose complexity is crucial for determining the complexity of PIE. We have shown that API is in NP (Proposition 29), and PIS is in CONP (Proposition 33). However we were not able to find matching lower bounds for these problems. We could show that API is COSIMPLE-H-SAT-hard (Theorem 30), and PIS is TRANS-HYP-hard (Corollary 34). Some interesting consequences of these results can be summed up as follows:

- If any of the problems API, or PIS turns out to be intractable, then unless P = NP, there cannot be an algorithm that solves PIE in output polynomial time (Propositions 28 and 32).
- Showing that any of the problems API or PIS is polynomial implies that the open problems TRANS-HYP and SIMPLE-H-SAT are also polynomial (Theorem 30, Corollary 34, [8]).
- Even if TRANS-HYP and SIMPLE-H-SAT turn out to be polynomial, API and PIS can still be intractable, thus it can still be the case that PIE is not solvable in output polynomial time.
- Even if API and PIS turn out to be polynomial, it can still be the case that PIE is not solvable in output polynomial time since Propositions 28 and 32 show only the other direction of the claim.

As future work in this direction, we are going to work on determining the exact complexity of the problems API and PIS. For API, we are going to investigate whether the hardness result [8] on hypergraph saturation for arbitrary graphs carries over to API on arbitrary formal contexts. For PIS, we are going to investigate the types of formal context where PIS and TRANS-HYP (and thus PIE and TRANS-ENUM) become computationally equivalent problems, and find out whether this type of formal contexts are natural in some applications, and how often they occur in practice.

In the present work we have also introduced the notion of a minimal pseudo-intent, which is a minimal pseudo-intent that does not contain any other pseudo-intents. We have shown that minimal pseudo-intents can be recognized in polynomial time (Section 6.1). In order to investigate whether minimal pseudo-intents can efficiently be enumerated, we have formulated a decision problem, namely ALL-MPI, which is the problem of checking whether a given set of minimal pseudo-intents are the set of all minimal pseudo-intents of a given context. Surprisingly it turned out that this problem is coNP-complete (Theorem 38), which implies that unless P = NP all minimal pseudo-intents of a context cannot be enumerated in output-polynomial time (Corollary 39). We have also shown that the problem of deciding whether a given pseudo-intent is lexicographically the last minimal pseudo-intent, namely LAST-MPI, is coNP-complete (Theorem 44).

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments that helped us to improve an earlier version of this work.

References

[1] M.A. Babin, S. Kuznetsov, Recognizing pseudo-intents is coNP-complete, in: M. Kryszkiewicz, S. Obiedkov (Eds.), Proceedings of the 2010 International Conference on Concept Lattices and their Applications, CLA 2010, CEUR Workshop Proceedings, vol. 672, 2010, pp. 294–301.
[2] C. Berge, Hypergraphs, Elsevier Science Publishers B.V., North Holland, 1989.
[3] P. Colomb, L. Nourine, About keys of formal context and conformal hypergraph, in: R. Medina, S.A. Obiedkov (Eds.), Proceedings of the 6th International Conference on Formal Concept Analysis, ICFCA 2008, in: Lecture Notes in Computer Science, vol. 4933, Springer-Verlag, 2008, pp. 140–149.
[4] F. Distel, Hardness of enumerating pseudo-intents in the lectic order, in: L. Kwuida, B. Sertkaya (Eds.), Proceedings of the 8th International Conference on Formal Concept Analysis, ICFCA 2010, in: Lecture Notes in Artificial Intelligence, vol. 5986, Springer-Verlag, 2010, pp. 124–137.
[5] W.F. Dowling, J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, Journal of Logic Programming 3 (1984) 267–284.
[6] V. Duquenne, The core of finite lattices, Discrete Mathematics 88 (1991) 133–147.
[7] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, Technical Report CD-TR 91/16, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1991.
[8] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, SIAM Journal on Computing 24 (6) (1995) 1278–1304.
[9] T. Eiter, G. Gottlob, Hypergraph transversal computation and related problems in logic and AI, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Proceedings of the European Conference on Logics in Artificial Intelligence, JELIA 2002, in: Lecture Notes in Computer Science, vol. 2424, Springer-Verlag, 2002, pp. 549–564.
[10] T. Eiter, G. Gottlob, K. Makino, New results on monotone dualization and generating hypergraph transversals, in: Proceedings on 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 14–22.
[11] T. Eiter, G. Gottlob, K. Makino, New results on monotone dualization and generating hypergraph transversals, SIAM Journal on Computing 32 (2) (2003) 514–537.
[12] T. Eiter, K. Makino, G. Gottlob, Computational aspects of monotone dualization: a brief survey, Discrete Applied Mathematics 156 (11) (2008) 2035–2049.
[13] M.L. Fredman, L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, Journal of Algorithms 21 (3) (1996) 618–628.
[14] B. Ganter, Two basic algorithms in concept analysis, Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany, 1984.
[15] B. Ganter, Two basic algorithms in concept analysis, in: L. Kwuida, B. Sertkaya (Eds.), Proceedings of the 8th International Conference on Formal Concept Analysis, ICFCA 2010, in: Lecture Notes in Artificial Intelligence, vol. 5986, Springer-Verlag, 2010, pp. 329–359. Reprint of [14].
[16] B. Ganter, R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer-Verlag, Berlin, Germany, 1999.
[17] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W.H. Freeman & Company, New York, NY, USA, 1990.
[18] A. Gély, R. Medina, L. Nourine, Y. Renaud, Uncovering and reducing hidden combinatorics in Guigues-Duquenne bases, in: B. Ganter, R. Godin (Eds.), Proceedings of the 3rd International Conference on Formal Concept Analysis, ICFCA 2005, in: Lecture Notes in Computer Science, vol. 3403, Springer-Verlag, 2005, pp. 235–248.
[19] A. Gély, L. Nourine, About the family of closure systems preserving non-unit implications in the Guigues-Duquenne base, in: R. Missaoui, J. Schmid (Eds.), Proceedings of the 4th International Conference on Formal Concept Analysis, ICFCA 2006, in: Lecture Notes in Computer Science, vol. 3874, Springer-Verlag, 2006, pp. 191–204.
[20] J.-L. Guigues, V. Duquenne, Familles minimales d'implications informatives resultant d'un tableau de données binaries, Mathématiques, Informatique et Sciences Humaines 95 (1986) 5–18.
[21] D. Gunopulos, R. Khardon, H. Mannila, H. Toivonen, Data mining, hypergraph transversals, and machine learning, in: Proceedings of the Sixteenth Symposium on Principles of Database Systems, PODS 97, 1997, pp. 209–216.
[22] M. Hagen, Algorithmic and computational complexity issues of Monet, Ph.D. Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2008.

[23] P. Janssen, L. Nourine, Minimum implicational basis for meet-semidistributive lattices, Information Processing Letters 99 (5) (2006) 199–202.
[24] D.S. Johnson, M. Yannakakis, C.H. Papadimitriou, On generating all maximal independent sets, Information Processing Letters 27 (3) (1988) 119–123.
[25] D.J. Kavvadias, C.H. Papadimitriou, M. Sideri, On Horn envelopes and hypergraph transversals, in: K.-W. Ng, P. Raghavan, N.V. Balasubramanian, F.Y.L. Chin (Eds.), 4th International Symposium on Algorithms and Computation, ISAAC'93, in: Lecture Notes in Computer Science, vol. 762, Springer-Verlag, 1993, pp. 399–405.
[26] D.J. Kavvadias, E.C. Stavropoulos, Checking monotone Boolean duality with limited nondeterminism, Technical Report TR2003/07/02, Computer Technology Institute, Patras, Greece, 2003.
[27] D.J. Kavvadias, E.C. Stavropoulos, Monotone Boolean dualization is in $\textsc{coNP}[\log^2 n]$, Information Processing Letters 85 (1) (2003) 1–6.
[28] L.G. Khachiyan, E. Boros, K.M. Elbassioni, V. Gurvich, K. Makino, On the complexity of some enumeration problems for matroids, SIAM Journal of Discrete Mathematics 19 (4) (2005) 966–984.
[29] S.O. Kuznetsov, On the intractability of computing the Duquenne–Guigues base, Journal of Universal Computer Science 10 (8) (2004) 927–933.
[30] S.O. Kuznetsov, S.A. Obiedkov, Counting pseudo-intents and #P-completeness, in: R. Missaoui, J. Schmid (Eds.), Proceedings of the 4th International Conference on Formal Concept Analysis, ICFCA 2006, in: Lecture Notes in Computer Science, vol. 3874, Springer-Verlag, Dresden, Germany, 2006, pp. 306–308.
[31] S.O. Kuznetsov, S.A. Obiedkov, Some decision and counting problems of the Duquenne–Guigues basis of implications, Discrete Applied Mathematics 156 (11) (2008) 1994–2003.
[32] E. Lawler, J. Lenstra, A.R. Kan, Generating all maximal independent sets: NP-hardness and polynomial time algorithms, SIAM Journal on Computing 9 (1980) 558–565.
[33] D. Maier, The Theory of Relational Databases, Computer Science Press, Maryland, 1983.
[34] R. Medina, C. Noyer, O. Raynauld, Efficient algorithms for clone items detection, in: R. Belohlavek, V. Snasel (Eds.), Proceedings of the 2005 International Workshop on Concept Lattices and their Applications, CLA 2005, in: CEUR Workshop Proceedings, vol. 162, 2005.
[35] S.A. Obiedkov, V. Duquenne, Attribute-incremental construction of the canonical implication basis, Annals of Mathematics and Artificial Intelligence 49 (1–4) (2007) 77–99.
[36] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, Massachusetts, 1994.
[37] S. Rudolph, Some notes on pseudo-closed sets, in: S.O. Kuznetsov, S. Schmidt (Eds.), Proceedings of the 5th International Conference on Formal Concept Analysis, ICFCA 2007, in: Lecture Notes in Computer Science, vol. 4390, Springer-Verlag, 2007, pp. 151–165.
[38] B. Sertkaya, Some computational problems related to pseudo-intents, in: S. Ferré, S. Rudolph (Eds.), Proceedings of the 7th International Conference on Formal Concept Analysis, ICFCA 2009, in: Lecture Notes in Artificial Intelligence, vol. 5548, Springer-Verlag, 2009, pp. 130–145.
[39] B. Sertkaya, Towards the complexity of recognizing pseudo-intents, in: F. Dau, S. Rudolph (Eds.), Proceedings of the 17th International Conference on Conceptual Structures, ICCS 2009, in: Lecture Notes in Computer Science, Springer-Verlag, 2009.
[40] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, A new algorithm for generating all maximal independent sets, SIAM Journal on Computing 6 (1977) 505–517.
[41] M. Wild, Optimal implicational bases for finite modular lattices, Quaestiones Mathematicae 23 (2000) 153–161.
[42] R. Wille, Restructuring lattice theory: an approach based on hierarchies of concepts, in: I. Rival (Ed.), Ordered Sets, Reidel, Dordrecht, Boston, 1982, pp. 445–470.
[43] R. Wille, Restructuring lattice theory: an approach based on hierarchies of concepts, in: S. Ferré, S. Rudolph (Eds.), Proceedings of the 7th International Conference on Formal Concept Analysis, ICFCA 2009, in: Lecture Notes in Artificial Intelligence, vol. 5548, Springer-Verlag, 2009, Reprint of [42].