On the Complexity of Computing Generators of Closed Sets

Miki Hermann[1] and Barış Sertkaya[2]

[1] LIX (CNRS, UMR 7161), École Polytechnique, 91128 Palaiseau, France
hermann@lix.polytechnique.fr
[2] Institut für Theoretische Informatik, TU Dresden, Germany
sertkaya@tcs.inf.tu-dresden.de

Abstract. We investigate the computational complexity of some decision and counting problems related to generators of closed sets fundamental in Formal Concept Analysis. We recall results from the literature about the problem of checking the existence of a generator with a specified cardinality, and about the problem of determining the number of minimal generators. Moreover, we show that the problem of counting minimum cardinality generators is #·coNP-complete. We also present an incremental-polynomial time algorithm from relational database theory that can be used for computing all minimal generators of an implication-closed set.

1 Introduction

Closed sets and pseudo-closed sets play an important rôle in Formal Concept Analysis (FCA) [5]. For instance, the sets closed under implications are fundamental to the attribute exploration algorithm [4]. In addition, pseudo-closed sets form the left-hand sides of the implications in the canonical implication base called the Duquenne-Guigues Base [7] of a formal context. As a result, many problems related to closed and pseudo-closed sets have been by now well investigated in the FCA community. For instance, there exist several polynomial-delay algorithms[3] that generate all concept intents of a formal context. Other computational problems related to pseudo-closed sets have been analyzed in [11,12,14].

Beside closed and pseudo-closed sets, generators of closed sets also play an important rôle in FCA. Inspite of this, as mentioned in [17], they have been paid little attention in the FCA community, especially computational problems related to them have not been well investigated. Different aspects of minimal generators have been investigated in the literature [3,17,23]. Valtchev et al. presented in [23] an efficient method for maintaining the set of minimal generators of all intents of a formal context upon increases in the object set of the underlying context. Nehmé et al. investigated in [17] the same problem in the dual setting. They presented a method for maintaining the set of minimal generators upon increases in the attribute set of the context. They characterized how the

[3] See [13] for a comprehensive list and a detailed comparison of these algorithms.

set of minimal generators changes when a new attribute is added to the context. Using this characterization they developed an efficient incremental algorithm for generating concept intents. Frambourg et al. worked in [3] on evolution of the the set of minimal generators during lattice assembly .

The present paper aims to given an overview of the computational complexity of some decision and counting problems on generators of closed sets. In particular we consider the two types of closed sets that are fundamental in FCA, namely concept intents and sets closed under a set of implications. Throughout the text, for the latter type of sets, we use the term implication-closed set. We recall results from the literature about the problem of checking the existence of a generator with a specified cardinality, and about the problem of determining the number of minimal generators. Moreover, we define a new problem about the second type of closed sets, namely the problem of determining the number of minimum cardinality generators, and show that this problem is #·coNP-complete, i.e., it is even more difficult than determining the number of minimal generators. We also point out that an incremental-polynomial time algorithm from relational database theory can be used for computing all minimal generators of an implication-closed set.

Our motivation for analyzing these problems is not only theoretical, but also practical. A good analysis of these problems can help to develop methods that support the expert during attribute exploration by making the implication questions "simpler". We know that the attribute exploration algorithm asks the smallest number of questions to the expert, i.e., none of the questions it asks is redundant. However, it might still be possible to shorten an implication question by removing redundant attributes from its premise and conclusion. Moreover, a good analysis of the problems related to generators of concept intents can help to develop efficient lattice construction and merge algorithms.

2 Counting Complexity

We assume that the reader has a basic knowledge of complexity theory. Additional information can be found in the book [19].

A *counting problem* is presented using a suitable *witness* function which for every input x returns a set of *witnesses* for x. Formally, a *witness* function is a function $A\colon \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are two alphabets, and $\mathcal{P}^{<\omega}(\Gamma^*)$ is the collections of all finite subsets of Γ^*. Every such witness function gives rise to the following *counting problem*: given a string $x \in \Sigma^*$, find the cardinality $|A(x)|$ of the *witness* set $A(x)$.

Complexity of counting problems was first investigated by Valiant in [21, 22]. For a systematical study and classification of counting problems he introduced the counting complexity class #P, defined as the class of functions counting the number of accepting paths of nondeterministic polynomial-time Turing machines. A typical member is the problem #SAT, counting the number of satisfying assignments to a propositional formula in conjunctive normal form. Valiant showed in [21, 22] that #SAT and many other problems are #P-complete.

Hemaspaandra and Vollmer introduced in [9] a predicate-based approach for defining higher counting complexity classes. In this approach, the counting complexity classes are denoted by $\#\cdot\mathcal{C}$.

Definition 1. *$\#\cdot\mathcal{C}$ is the class of all counting problems whose witness function A satisfies the following conditions:*

(i) There is a polynomial $p(n)$ such that every $x \in \Sigma^*$ and every $y \in A(x)$ satisfy the relation $|y| \leq p(|x|)$;

(ii) The decision problem "given x and y, does y belong to $A(x)$?" is in $\mathcal{C}$.

Completeness of the problems in #P is often proved by using *parsimonious reductions*, which are polynomial-time reductions preserving the number of solutions by establishing a bijection between the solution sets of the problems. There are, however, two shortcoming of parsimonious reductions. First, they are not powerful enough, since they represent a particular case of many-one reductions, whereas Valiant was obliged to use Turing reductions in [21, 22] to be able to prove #P-completeness of several problems like #PERMANENT or #PERFECT MATCHINGS. Second, even if the many-one reduction is powerful enough for proving completeness, there does not need to exist a one-to-one correspondence between the solutions of the reduced problems. On the other hand, Turing reductions turned out to be too powerful, since as it was proved in [20], they collapse all counting classes $\#\cdot\Sigma_k\mathrm{P}$ and $\#\cdot\Pi_k\mathrm{P}$ to #P.

In order to overcome this problem, Durand et al. introduced in [2] a new kind of reductions called *subtractive reduction*, under which #P and the higher classes $\#\cdot\Pi_k\mathrm{P}$ for each $k \in \mathbb{N}$ are closed. A subtractive reduction between counting problems first overcounts the number of solutions and then carefully subtracts any surplus. It is formally defined as follows.

Definition 2. *Let Σ, Γ be two alphabets and let $\#\cdot A$ and $\#\cdot B$ be two counting problems determined by the binary relations A and B between strings from Σ to Γ. We say that $\#\cdot A$ reduces to $\#\cdot B$ via a* strong subtractive reduction *if there exist two polynomial-time computable functions f and g such that for every string $x \in \Sigma^*$ the following conditions hold.*

1. $B(f(x)) \subseteq B(g(x))$;
2. $|A(x)| = |B(g(x))| - |B(f(x))|$.

A subtractive reduction *is a transitive closure of strong subtractive reductions.*

Parsimonious reductions constitute a special case of subtractive reductions with $B(f(x)) = \emptyset$. In [2] it was pointed out that subtractive reductions are well-suited tools to study the higher counting complexity classes $\#\cdot\Sigma_k\mathrm{P}$ and $\#\cdot\Pi_k\mathrm{P}$.

3 Generators of Concept Intents

We assume that the reader is familiar with the theory of FCA. We briefly mention the necessary basic notions and refer the reader to the standard textbook [5] for additional information. In the present section we shortly recall the notion of generators of a concept intent, and some well-known computational problems about them.

Definition 3. *Let $\mathbb{K} = (G, M, I)$ be a formal context and $C \subseteq M$ be a concept intent, i.e., $C'' = C$. The subset $D \subseteq C$ is a minimal generator of C under $(\cdot)''$ if $D'' = C$ holds and D is subset-minimal, i.e., for all $E \subsetneq D$ we have $E'' \subsetneq C$.*

We first recall the computational complexity of checking whether a concept intent has a generator of cardinality less than or equal to a specified size. It is well-known that the following problem is NP-complete.

Problem: INTENT GENERATOR
Input: A formal context $\mathbb{K} = (G, M, I)$, the intent D of a formal concept (C, D) from $\mathbb{K}$, and a positive integer $m \leq |A|$.
Question: Is there a subset $Q \subseteq D$ of cardinality less than or equal to m that generates D, i.e., is there a $Q \subseteq D$ such that $Q'' = D$ and $|Q| \leq m$?

Frambourg et al. mentioned in [3] that the number of minimal generators of an intent can be exponential in the size of the context. Apart from this exponential bound, it is common folklore that the following problem is #P-complete.

Problem: #MINIMAL INTENT GENERATOR
Input: A formal context $\mathbb{K} = (G, M, I)$ and the intent D of a formal concept (C, D) in $\mathbb{K}$.
Output: Number of all subset-minimal intent generators of D with respect to the closure operator $(\cdot)''$, i.e., $|\{Q \subseteq D \mid Q'' = D \wedge \forall P \subsetneq Q,\ P'' \neq D\}|$.

4 Generators of Implication-Closed Sets

In the present section we first shortly recall the notion of minimal generators of an implication-closed set, and some well-known computational problems about minimal generators. Later we define a new problem about minimal generators, and work its computational complexity.

Definition 4. *Let $\mathcal{L}$ be a set of implications on a finite attribute set A and $P \subseteq A$ be closed with respect to $\mathcal{L}$, i.e., $\mathcal{L}(P) = P$. The subset $Q \subseteq P$ is a minimal generator of P under $\mathcal{L}$ if $\mathcal{L}(Q) = P$ holds and Q is subset-minimal, i.e., for all $R \subsetneq Q$ we have $\mathcal{L}(R) \subsetneq P$.*

Minimal generators appear in the literature under different names in various fields. For instance, in relational databases they are called minimal keys, and various properties of them have been considered in the literature. In order to make this connection clear, let us briefly recall some basic notions of relational databases.

4.1 Connection to Relational Databases

Functional dependencies are a way of expressing constraints on data in relational databases [16]. Informally, a functional dependency occurs when the values of a tuple on one set of attributes uniquely determine the values on another set

of attributes. Formally, given a relation R and a set of attribute names A, a *functional dependency* is a pair of sets $X, Y \subseteq A$ written as $X \to Y$. The relation R *satisfies* the functional dependency $X \to Y$ if the tuples with equal X-values also have equal Y-values. In this case we say that the set of attributes X *functionally determine* the set of attributes Y.

Another important concept in relational databases is the notion of a key. Given a relation R on the attribute set A, a set $K \subseteq A$ is called a *key* of R if K functionally determines A. It is called a *minimal key* if no proper subset of it is a key. Alternatively, given a set of functional dependencies F that are satisfied by R, a set $K \subseteq A$ is called a key of the *relational system* $\langle A, F \rangle$ if $K \to A$ can be inferred from F by using Armstrong's axioms [1]. In practical applications, it is important to find "small" keys of a given relation. Lucchesi and Osborn analyzed in [15] how difficult it is to check whether a given relation has a key of cardinality bounded by a specified size. This problem is known as the MINIMUM CARDINALITY KEY problem (see problem [SR26] in [6]).

Problem: MINIMUM CARDINALITY KEY
Input: A set A of attribute names, a collection F of functional dependencies, and a positive integer $m \in \mathbb{N}$.
Question: Is there a key of cardinality m or less for the relational system $\langle A, F \rangle$?

Lucchesi and Osborn proved in [15] that MINIMUM CARDINALITY KEY is NP-complete. It is well-known that minimal generators of a closed set are the minimal keys of the subrelation defined by this closed set. Based on this observation, it is part of common folklore that the following problem is also NP-complete.

Problem: MINIMUM CARDINALITY GENERATOR
Input: A set A of attribute names, a set $\mathcal{L}$ of implications on A, an $\mathcal{L}$-closed subset P of A, and a positive integer $m \leq |A|$.
Question: Is there a subset $Q \subseteq P$ of cardinality $|Q| \leq m$ that generates P under $\mathcal{L}$, i.e., is there a $Q \subseteq P$ such that $\mathcal{L}(Q) = P$ and $|Q| \leq m$?

4.2 Counting Minimal Generators

Osborn showed in [18] that the number of minimal keys for a relational system $\langle A, F \rangle$ can be exponential in $|A|$. Moreover, Gunopulos et al. proved in [8] that the problem of determining the number of minimal keys of a relational system is #P-complete. Due to the correspondence between minimal keys and minimal generators of a closed set, it is also well-known that the number of minimal generators can be exponential in the size of the attribute set, and that the following counting problem is #P-complete.

Problem: #MINIMAL GENERATOR
Input: A set A of attribute names, a set $\mathcal{L}$ of implications on A, and an $\mathcal{L}$-closed subset P of A.
Output: Number of all subset-minimal generators of P under $\mathcal{L}$.

Algorithm 1 Minimal generator

Input: Implications $\mathcal{L}$ on the attribute set A and a subset $P \subseteq A$ such that $\mathcal{L}(P) = P$
Output: A minimal generator Q of P

```
1: Q ← P
2: for all m ∈ P do
3:    if L(Q ∖ {m}) = P then
4:       Q ← Q ∖ {m}
5:    end if
6: end for
```

4.3 Finding All Minimal Generators

In some cases, it might not be enough to find only one minimal generator of an implication-closed set. For instance during attribute exploration it might be useful to show the expert different minimal generators of the premise and conclusion of the implication question for better understandability. The expert might want to browse among them to find a shortened version of the question which is most comprehensible to him. In the sequel we are going to investigate the problem of determining all minimal generators of a closed set.

Lucchesi and Osborn presented in [15] an algorithm to determine all minimal keys of a given relation. Given a set of attributes R and a set of functional dependencies F, the algorithm returns the set of all minimal keys for the relational system $\langle R, F\rangle$. Below we present an adaptation of this algorithm to find all minimal generators of a given implication-closed set. The algorithm is based on the following property shown in [15]. Here we formulate the property in terms of implications and minimal generators, and leave out its proof.

Lemma 5. *Let $\mathcal{L}$ be a set of implications on the attributes A and $\mathcal{G}$ be a nonempty set of minimal generators for a given $P \subseteq A$ under $\mathcal{L}$. The complement set $2^P \setminus \mathcal{G}$ contains a minimal generator if and only if $\mathcal{G}$ contains a minimal generator G and $\mathcal{L}$ contains an implication $L \to R$, such that $L \cup R \cup G \subseteq P$ holds and $L \cup (G \setminus R)$ does not include any minimal generator from $\mathcal{G}$.*

Lemma 5 assumes the existence of a nonempty set of minimal generators, thus the algorithm following from the lemma needs one minimal generator before it can proceed to find all other minimal generators. It is not difficult to find one minimal generator of a given implication-closed set P. We can start with P, iterate over all elements of P, and remove an element if the remaining set still generates P. Algorithm 1 implements this idea. It determines a minimal generator of a given set of attributes P closed under a given set of implications $\mathcal{L}$. Algorithm 1 terminates since P is finite. Upon termination, Q is a minimal generator of P since it does not contain any redundant attributes. For checking whether $Q \setminus \{m\}$ generates P we can use the well-known implicational closure algorithm LinClosure from [16]. The LinClosure algorithm runs in time $O(|\mathcal{L}|\,|A|)$. Algorithm 1 makes at most $|A|$ iterations of LinClosure and therefore it runs in time $O(|\mathcal{L}|\,|A|^2)$.

Algorithm 2 All minimal generators

Input: Set of implications $\mathcal{L}$ on the attribute set A and an $\mathcal{L}$-closed set $P \subseteq A$
Output: All minimal generators $\mathcal{G}$ of P

```
1: G ← {MinGen(P, L)}                          {Initial set of minimal generators}
2: for all G ∈ G do
3:    for all L → R ∈ L such that L ∪ R ∪ G ⊆ P do
4:       S ← L ∪ (K ∖ R)
5:       flag ← true
6:       for all H ∈ G do
7:          if H ⊆ S then
8:             flag ← false
9:          end if
10:      end for
11:      if flag then
12:         G ← G ∪ {MinGen(S, L)}
13:      end if
14:   end for
15: end for
```

Now that we have an algorithm to determine one minimal generator, we can proceed with the algorithm determining the set of all minimal generators of an implication-closed set.

Algorithm 2 terminates, since $\mathcal{G}$ and $\mathcal{L}$ are both finite. Following Lemma 5, upon termination of the algorithm the set $\mathcal{G}$ contains all minimal generators of the given set of attributes P under $\mathcal{L}$. Let $|\mathcal{L}| = \ell$, $|\mathcal{G}| = g$, and $|P| = p$ be the cardinalities of the corresponding sets. The algorithm runs in time $O(\ell g(p + gp)) + O(gm)$, where m is the complexity of Algorithm 1. Hence Algorithm 2 has time complexity $O(\ell gp(g + p))$. Note that the algorithm finds minimal generators in *incremental polynomial time*, which is a notion introduced in [10] for analyzing the performance of algorithms that generate all solutions of a problem. An algorithm is said to run in incremental polynomial time if given an input and a prefix of the set of solutions (say, a closed set and a collection of the first k minimal generators), it finds another solution, or determines that none exists, in time polynomial in the combined sizes of the input and the given prefix. For finding a minimal generator, Algorithm 2 needs to perform at most $g\ell p(g + p)$ operations, which is polynomial in the size of the input, i.e., in the size of $\mathcal{L}$ and P, as well as polynomial in the size of the already found minimal generators $\mathcal{G}$.

Another notion introduced in [10] for analyzing algorithms that enumerate solutions is polynomial delay. An algorithm is said to run with *polynomial delay* if the delay until the first solution is written, as well as thereafter the delay between any two consecutive solutions, is bounded by a polynomial in the size of the input. Polynomial delay is a stronger notion than incremental polynomial time, i.e., if an algorithm runs with polynomial delay it is also runs in incremental polynomial time. To the best of our knowledge, there is no polynomial delay

algorithm that finds all minimal keys of a relation, which is equivalent to finding all minimal generators of an attribute set closed under a set of implications.

4.4 Counting Minimum Cardinality Generators

In this section we consider a modified version of the #MINIMAL GENERATOR. For this problem, we slightly change the notion of "generates" as follows. For a given set $\mathcal{L}$ of implications on an attribute set A, and an $\mathcal{L}$-closed set $P \subseteq A$, we say that a $Q \subseteq A$ is a *minimum cardinality generator* of P if $\mathcal{L}(Q) \smallsetminus Q = P$ holds and no subset of A with smaller cardinality satisfies this property. In other words, we require that P should be the "new consequences" of closing Q under $\mathcal{L}$ and that no set with smaller cardinality can have this property. It turns out that the problem of counting such sets is #·coNP-complete, which means that it is even harder than the #MINIMAL GENERATOR problem.

Problem: #MINIMUM CARDINALITY GENERATOR
Input: A set A of attribute names, a set $\mathcal{L}$ of implications on A, an $\mathcal{L}$-closed subset P of A.
Question: Number of all minimum cardinality generators of P under $\mathcal{L}$, i.e., number of the subsets $Q \subseteq A$ such that $\mathcal{L}(Q) \smallsetminus Q = P$ and no other subset $R \subseteq A$ with $|R| < |Q|$ satisfies the condition $\mathcal{L}(R) \smallsetminus R = P$.

Theorem 6. #MINIMUM CARDINALITY GENERATOR *is* #·coNP*-complete.*

Proof. The problem is clearly in #·coNP what can be shown as follows. Given a set of attributes Q, we have to check (i) whether Q generates P, and if so (ii) whether there is another generator R with $|R| < |Q|$. The first test can be done in polynomial time using a closure algorithm based on the reachability algorithm for graphs. The second test, which dominates the overall complexity, can be done by a coNP-algorithm. Indeed, checking whether Q is *not* a minimum cardinality generator can be done by the following NP-algorithm: Guess a subset of attributes $R \subseteq A$ such that $|R| < |Q|$ and check if R generates P. Again, checking if R generates P can be done in polynomial time, thus checking whether Q is a minimum cardinality generator can be done in coNP and counting such sets can be done in #·coNP.

We show the #·coNP-hardness by a strong subtractive reduction from the problem #Π_1SAT. #Π_1SAT is #·coNP-complete according to [2]. Consider an instance of the #Π_1SAT problem given by a formula $\varphi(X) = \forall Y\ \psi(X, Y)$ where $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_l\}$ are disjoint sets of variables. Without loss of generality we can assume that $\psi(X, Y)$ is in 3DNF, i.e., it is of the form $C_1 \vee \cdots \vee C_n$ where each C_i is of the form $C_i = l_{i1} \wedge l_{i2} \wedge l_{i3}$, and the l_{ij}'s are propositional literals over $X \cup Y$.

Let $x'_1, \ldots, x'_k$, $q_1, \ldots, q_k$, $y'_1, \ldots, y'_l$, $r_1, \ldots, r_l$, $g_1, \ldots, g_n$, u denote fresh pairwise distinct variables and let us regroup them in the sets $X_{=}\{x'_1, \ldots, x'_k\}$, $Y_{=}\{y'_1, \ldots, y'_l\}$, $Q_1 = \{q_1, \ldots, q_k\}$, $R_1 = \{r_1, \ldots, r_l\}$, and $G = \{g_1, \ldots, g_n\}$.

We define two instances of the minimum cardinality generator problem. The first problem $\mathbb{P}_1$ is defined as follows:

$$\begin{aligned}
A_1 &= A = X \cup X_1 \cup Y \cup Y_1 \cup Q_1 \cup R_1 \cup G \cup \{u\} \\
P_1 &= Q_1 \cup R_1 \cup G \\
\mathcal{L}_1 &= \{\{x_i, x_i'\} \rightarrow A,\ x_i \rightarrow q_i,\ x_i' \rightarrow q_i \mid 1 \leq i \leq k\} \ \cup \\
&\quad \{\{y_i, y_i'\} \rightarrow A,\ y_i \rightarrow r_i,\ y_i' \rightarrow r_i \mid 1 \leq i \leq l\} \ \cup \\
&\quad \{z_{ij} \rightarrow g_i \mid 1 \leq i \leq n \quad \text{and} \quad 1 \leq j \leq 3\}
\end{aligned}$$

where, for $1 \leq s \leq k$ and $1 \leq t \leq l$, z_{ij} is in one of the forms $x_s, x_s', y_t,$ or, y_t' depending on whether the literal l_{ij} in C_i is in one of the forms $\neg x_s, x_s, \neg y_t$, or y_t, respectively. In other words, z_{ij} encodes the negation of l_{ij}. Now we define the second problem $\mathbb{P}_2$.

$$A_2 = A, \quad P_2 = P_1, \quad \mathcal{L}_2 = \mathcal{L}_1 \cup \{\{y_1, \ldots, y_l\} \rightarrow g_i \mid 1 \leq i \leq n\}.$$

Now let $\mathcal{A}(\varphi)$ denote the set of all satisfying truth assignments of a $\#\Pi_1$SAT-formula φ and let $\mathcal{B}(\mathbb{P})$ denote the set of all solutions of a minimum cardinality generator problem $\mathbb{P}$. We claim that the following holds:

$$\mathcal{B}(\mathbb{P}_1) \subseteq \mathcal{B}(\mathbb{P}_2) \quad \text{and} \quad |\mathcal{A}(\varphi)| = |\mathcal{B}(\mathbb{P}_2)| - |\mathcal{B}(\mathbb{P}_1)|\,.$$

Consider the problem $\mathbb{P}_1$. Solutions of $\mathbb{P}_1$, i.e., minimum cardinality generators of P_1 satisfy the following 3 conditions: (1) An attribute q_i can be generated only in two ways, by the implication $x_i \rightarrow q_i$ or by the implication $x_i' \rightarrow q_i$. So a solution of $\mathbb{P}_1$ contains one of x_i and x_i'. Moreover, it cannot contain both of them due to the implication $\{x_i, x_i'\} \rightarrow A$, since this implication would also generate the attribute u, and u is not contained in P_1. This means, for each $1 \leq i \leq k$ a solution of $\mathbb{P}_1$ contains either x_i or x_i' in order to be able to generate the q_i's. (2) Similarly, it also contains either y_i or y_i' for each $1 \leq i \leq l$ in order to be able to generate the r_i's. (3) In addition to these, in order to be able to generate an attribute g_i, a solution contains at least one attribute that encodes the negation of a literal occurring in the implicant C_i. In order be able to generate all g_i's, a solution contains at least one such attribute for each implicant C_i. Subsets of A that satisfy these 3 conditions are solutions of $\mathbb{P}_1$. Each such subset has exactly the size $|X| + |Y| = k + l$. Moreover, they are the only solutions of $\mathbb{P}_1$, since any subset of A that has cardinality less than $k + l$ fails to generate at least one attribute in P_1. Conditions (1) and (2) enforce a solution to be a truth assignment over $X \cup Y$. Condition (3) enforces this truth assignment to contain the negation of at least one literal in every implicant, i.e., it enforces this truth assignment to falsify the formula $\psi(X, Y)$.

Consider now the problem $\mathbb{P}_2$. Each solution of $\mathbb{P}_1$ is also a solution of $\mathbb{P}_2$ since $P_2 = P_1$ and $\mathcal{L}_2$ contains all implications from $\mathcal{L}_1$. In addition to the implications from $\mathcal{L}_1$, $\mathcal{L}_2$ also contains implications of the form $\{y_1, \ldots, y_l\} \rightarrow g_i$ for each $1 \leq i \leq n$. These new implications give rise to the following new solutions. Like the solutions of $\mathbb{P}_1$, in order to be able to generate the q_i's and r_i's, they satisfy

the conditions (1) and (2) mentioned above. In order to be able to generate the g_i's, they contain every y_i for each $1 \leq i \leq l$. In other words, these new solutions are truth assignments over $X \cup Y$ that set every $y_1, \ldots, y_l$ to *true*.

Based on the above descriptions, $\mathcal{B}(\mathbb{P}_1)$ is the set of truth assignments that *falsify* $\psi(X, Y)$ and $\mathcal{B}(\mathbb{P}_2)$ is the set of truth assignments that *falsify* $\psi(X, Y)$, plus the set of truth assignments that set every $y_1, \ldots, y_l$ to *true*. Obviously, the claim $\mathcal{B}(\mathbb{P}_1) \subseteq \mathcal{B}(\mathbb{P}_2)$ is satisfied. Moreover, the difference $\mathcal{B}(\mathbb{P}_1) \setminus \mathcal{B}(\mathbb{P}_2)$ is the set of truth assignments that set every $y_1, \ldots, y_l$ to *true* and at the same time *satisfy* $\psi(X, Y)$ (since by taking the set difference from $\mathcal{B}(\mathbb{P}_1)$ we remove the truth assignments that falsify $\psi(X, Y)$). In other words, this set contains the models of $\psi(X, Y)$ such that all Y values are fixed by setting them to *true*. This set has exactly the same cardinality as the set of models of $\varphi(X) = \forall Y\ \psi(X, Y)$, thus the other claim $|\mathcal{A}(\varphi)| = |\mathcal{B}(\mathbb{P}_2)| - |\mathcal{B}(\mathbb{P}_1)|$ holds. □

5 Concluding Remarks

We analyzed some decision and counting problems related to generators of closed sets fundamental in FCA, namely concept intents and implication-closed sets. We have shown that the problem of checking the existence of a generator of cardinality less than or equal to a specified size is NP-complete and the problem of determining the number of minimal generators is #P-complete. Moreover, we have shown that the problem of determining the number of minimum cardinality generators is #·coNP-complete, i.e., it is even more difficult than counting minimal generators. We have also given an incremental-polynomial time algorithm from relational databases that can be used for computing all minimal generators of an implication-closed set. We also explicitly establish a connection between the corresponding results on relational databases and those on minimal keys of a relation, which were known before but have never explicitly been mentioned in the FCA literature.

It is not surprising to see that the mentioned problems about generators of concept intents and generators of implication-closed sets are of the same complexity. In fact, the closure operator induced by a formal context and the closure operator induced by the set of implications that are valid in this formal context coincide. That is, one can easily transfer these results from one case to the other.

References

1. W. W. Armstrong. Dependency structures of data base relationships. In J. L. Rosenfeld, editor, *Proceedings 6th Information Processing Conference (IFIP '74), Stockholm (Sweden)*, pages 580–583. North-Holland, 1974.
2. A. Durand, M. Hermann, and P. G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science*, 340(3):496–513, 2005.
3. C. Frambourg, P. Valtchev, and R. Godin. Merge-based computation of minimal generators. In F. Dau, M.-L. Mugnier, and G. Stumme, editors, *Proceedings 13th ICCS, Kassel (Germany)*, LNCS 3596, pages 181–194. Springer, 2005.

4. B. Ganter. Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Germany, 1984.
5. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
6. M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co, 1979.
7. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives resultant d'un tableau de données binaries. *Mathématiques, Informatique et Sciences Humaines*, 95:5–18, 1986.
8. D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. Sewak Sharma. Discovering all most specific sentences. *ACM Transactions on Database Systems*, 28(2):140–174, 2003.
9. L. A. Hemaspaandra and H. Vollmer. The satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News, Complexity Theory Column 8*, 26(1):2–13, March 1995.
10. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
11. S. O. Kuznetsov. On computing the size of a lattice and related decision problems. *Order*, 18(4):313–321, 2001.
12. S. O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *Journal of Universal Computer Science*, 10(8):927–933, 2004.
13. S. O. Kuznetsov and S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.
14. S. O. Kuznetsov and S. O. Obiedkov. Counting pseudo-intents and #P-completeness. In R. Missaoui and J. Schmid, editors, *Proceedings 4th ICFCA, Dresden (Germany)*, LNCS 3874, pages 306–308. Springer, February 2006.
15. C. L. Lucchesi and S. L. Osborn. Candidate keys for relations. *Journal of Computer and System Science*, 17(2):270–279, 1978.
16. D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.
17. K. Nehmé, P. Valtchev, M. H. Rouane, and R. Godin. On computing the minimal generator family for concept lattices and icebergs. In B. Ganter and R. Godin, editors, *Proceedings of the 3rd ICFCA, Lens (France)*, LNCS 3403, pages 192–207. Springer, 2005.
18. S. L. Osborn. *Normal Forms for Relational Data Bases*. PhD thesis, University of Waterloo, Canada, 1977.
19. C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
20. S. Toda and O. Watanabe. Polynomial-time 1-Turing reductions from #PH to #P. *Theoretical Computer Science*, 100(1):205–221, 1992.
21. L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.
22. L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
23. P. Valtchev, R. Missaoui, and R. Godin. Formal concept analysis for knowledge discovery and data mining: The new challenges. In P. W. Eklund, editor, *Proceedings 2nd ICFCA*, LNCS 2961, pages 352–371. Springer, 2004.