Listing Closed Sets of Strongly Accessible Set Systems with Applications to Data Mining[☆]

Mario Boley[a,*], Tamás Horváth[b,a], Axel Poigné[a], Stefan Wrobel[a,b]

[a] *Fraunhofer IAIS, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany*
[b] *Department of Computer Science, University of Bonn, Germany*

Abstract

We study the problem of listing all closed sets of a closure operator σ that is a partial function on the power set of some finite ground set E, i.e., $\sigma : \mathcal{F} \to \mathcal{F}$ with $\mathcal{F} \subseteq \mathcal{P}(E)$. A very simple divide-and-conquer algorithm is analyzed that correctly solves this problem if and only if the domain of the closure operator is a strongly accessible set system. Strong accessibility is a strict relaxation of greedoids as well as of independence systems. This algorithm turns out to have delay $O(|E|\,(T_{\mathcal{F}} + T_\sigma + |E|))$ and space $O(|E| + S_{\mathcal{F}} + S_\sigma)$, where $T_{\mathcal{F}}$, $S_{\mathcal{F}}$, T_σ, and S_σ are the time and space complexities of checking membership in $\mathcal{F}$ and computing σ, respectively. In contrast, we show that the problem becomes intractable for accessible set systems. We relate our results to the data mining problem of listing all support-closed patterns of a dataset and show that there is a corresponding closure operator for all datasets if and only if the set system satisfies a certain confluence property.

Key words: algorithms, listing, closure operator, data mining

1. Introduction

The problem of listing all closed sets (i.e., fixpoints) of a given arbitrary closure operator $\sigma : \mathcal{P}(E) \to \mathcal{P}(E)$, where $\mathcal{P}(E)$ denotes the power set of some finite set E, is well studied in different areas (see, e.g., [1, 2, 4]). In this work, we consider the problem generalization allowing closure operators that are only defined on some restricted domain, i.e., $\sigma : \mathcal{F} \to \mathcal{F}$ with $\mathcal{F} \subseteq \mathcal{P}(E)$. In addition, we investigate the relation of this problem to listing the family of all *support-closed* patterns of a dataset as defined in data mining. Given a transactional database, a set is called support-closed if all its supersets are contained in strictly less transactions than itself. We discuss some motivating examples in Section 7.

[☆] An early version of this paper appeared in the *Proc. of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases* (PKDD), LNAI 4702, pp. 382-389, Springer-Verlag, Heidelberg, 2007.

[*] Corresponding author

Email addresses: mario.boley@iais.fraunhofer.de (Mario Boley), tamas.horvath@iais.fraunhofer.de (Tamás Horváth), axel.poigne@iais.fraunhofer.de (Axel Poigné), stefan.wrobel@iais.fraunhofer.de (Stefan Wrobel)

For a set system $(E, \mathcal{F})$ with closure operator σ let n denote the size of E and N the number of closed sets. Moreover, let $T_{\mathcal{F}}$, $S_{\mathcal{F}}$, T_σ, and S_σ be the time and space complexity of checking membership in $\mathcal{F}$ and computing σ, respectively. In this paper we present the following results:

(i) In Sections 3 and 4 we consider a simple divide-and-conquer algorithm and show that it correctly lists all closed sets of *strongly accessible* set systems with delay $O(n(T_{\mathcal{F}} + T_\sigma + n))$, hence, total time $O(Nn(T_{\mathcal{F}} + T_\sigma + n))$, and space $O(n + T_{\mathcal{F}} + T_\sigma)$. Strong accessibility means that every $Y \in \mathcal{F}$ can be reached from all $X \subset Y$ with $X \in \mathcal{F}$ via augmentations with single elements "inside $\mathcal{F}$". This is a strict relaxation of independence systems as well as of greedoids and can be thought of as an abstract generalization of connectivity in the sense that the family of all connected vertex sets of a graph always forms a strongly accessible set system. The algorithm also provides an *algorithmic characterization* of strongly accessible set systems because it is correct for *all* closure operators of an input set system if and only if that set system is strongly accessible. As we discuss in Section 8, this is a difference to related approaches like our former algorithm [8] or an algorithm described by Arimura and Uno [9].

(ii) In Section 5 we show that the problem becomes intractable for the class of *accessible* set systems. Specifically, we prove a lower bound of $\Omega(2^{n/4})$ on the worst-case number of closure and membership computations that has to be performed by any correct algorithm accessing the input only via these two operations. This bound holds even if the problem is restricted to instances with a constant number of closed sets. In particular this shows that no output polynomial time algorithm exists for that task.

(iii) In Section 6 we show that support-closedness for all datasets is induced by a closure operator if and only if the set system satisfies a certain *confluence* property. Moreover, a corresponding closure operator can be computed efficiently if its domain is strongly accessible. In conjunction with result (i) we have an $O\big(n^2\,(|\mathcal{D}| + nT_{\mathcal{F}})\big)$ delay and $O(n + S_{\mathcal{F}})$ space algorithm for listing the support-closed patterns of confluent and strongly accessible set systems $(E, \mathcal{F})$ with respect to a given dataset $\mathcal{D}$. This constitutes a fairly general sufficiency criterion for the tractability of listing all support-closed patterns of a dataset. In contrast, if there is no corresponding closure operator, the problem turns out to be hard even for independence systems.

2. Definitions

A (finite) *set system* is an ordered pair $(E, \mathcal{F})$, where E is some (finite) set, called *ground set*, and $\mathcal{F} \subseteq \mathcal{P}(E)$. In this paper we consider only finite non-empty set systems. Let $(E, \mathcal{F})$ be a set system. A mapping $\sigma\colon \mathcal{F} \to \mathcal{F}$ is called a *closure operator* if it satisfies for all $X, Y \in \mathcal{F}$ that

- $X \subseteq \sigma(X)$ (*extensivity*),
- $X \subseteq Y \Rightarrow \sigma(X) \subseteq \sigma(Y)$ (*monotonicity*), and
- $\sigma(X) = \sigma(\sigma(X))$ (*idempotence*).

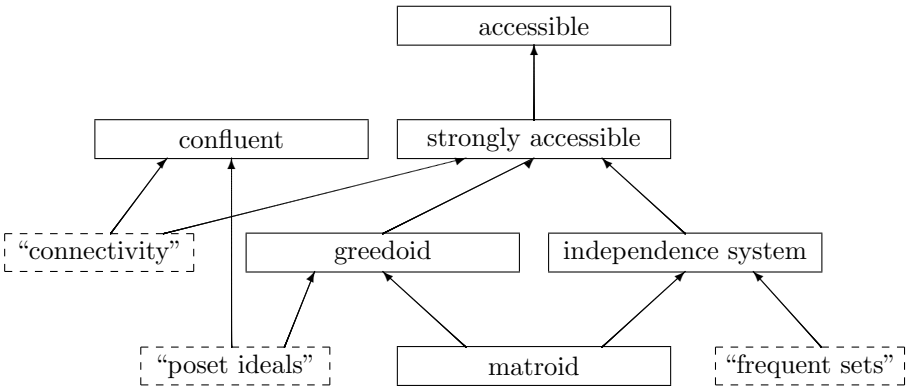

Figure 1: Relations among the introduced classes of set systems and application examples.

A set $F \in \mathcal{F}$ is called *closed* if it is a fixpoint of σ, i.e., if $\sigma(F) = F$. The family of closed elements of $\mathcal{F}$ is denoted by $\sigma(\mathcal{F})$, i.e., $\sigma(\mathcal{F}) = \{F \in \mathcal{F} \colon \sigma(F) = F\}$. Note that we defined the domain of the closure operator as some subset of $\mathcal{P}(E)$, and not $\mathcal{P}(E)$. Thus, in general, σ does not induce a *closure system* on $\mathcal{F}$: since $\mathcal{F}$ is not necessarily closed under intersection, neither is $\sigma(\mathcal{F})$.

The main computational problem studied in this work can then be formalized as follows.

Problem 1 (LIST-CLOSED-SETS**).** *Given* a set system $(E, \mathcal{F})$ with $\emptyset \in \mathcal{F}$ and a closure operator $\sigma : \mathcal{F} \to \mathcal{F}$, *list* the elements of $\sigma(\mathcal{F})$.

We assume that the closure operator as well as the set system are given implicitly by a closure oracle respectively a *membership oracle*, i.e., a boolean-valued function that, for every $F \subseteq E$, returns "true" if and only if $F \in \mathcal{F}$. The computational complexity of algorithms for Problem 1 will be investigated with respect to those of the membership and closure oracles as well as to the size of the ground set $|E| = n$. Since $|\sigma(\mathcal{F})|$ can in general be as large as 2^n (e.g., $\mathcal{F} = \mathcal{P}(E)$ and σ is the identity operator), there is no algorithm solving LIST-CLOSED-SETS in time polynomial in n. Thus, one aims for a good time bound *per closed set* and particularly a good bound on the *delay*, i.e., the maximum time between the output of two successive sets (see, e.g., [6]).

We investigate the properties of Problem 1 with respect to different structural assumptions on the input set system. A (non-empty) set system $(E, \mathcal{F})$ is called

- *accessible* if for all $X \in \mathcal{F} \setminus \{\emptyset\}$ there is an $e \in X$ such that $X \setminus \{e\} \in \mathcal{F}$,
- an *independence system* if $Y \in \mathcal{F}$ and $X \subseteq Y$ together imply $X \in \mathcal{F}$,
- a *greedoid* if it is accessible and satisfies the augmentation property, i.e., for all $X, Y \in \mathcal{F}$ with $|X| < |Y|$, there is an element $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{F}$, and
- a *matroid* if it is a greedoid and an independence system.

In addition, we introduce two new classes of set systems that are related to Problem 1 as we will show in subsequent sections.

Definition 1 (Strongly Accessible). A set system $(E, \mathcal{F})$ is called *strongly accessible* if it is accessible and for all $X, Y \in \mathcal{F}$ with $X \subset Y$, there is an $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{F}$.

By definition, strongly accessible set systems are also accessible. Moreover, it is easy to see that strong accessibility generalizes independence systems as well as greedoids. The next class of set systems does not stand in any containment relation with those given previously.

Definition 2 (Confluent). A set system $(E, \mathcal{F})$ is called *confluent* if for all $I, X, Y \in \mathcal{F}$ with $\emptyset \neq I \subseteq X$ and $I \subseteq Y$ it holds that $X \cup Y \in \mathcal{F}$.

The relations among all introduced set system classes are illustrated in Figure 1 along with the application problems considered in Section 7.

3. Divide-And-Conquer Closed Set Listing— An Algorithmic Characterization of Strong Accessibility

Algorithm 1 Divide & Conquer Closed Set Listing

Input : finite set system $(E, \mathcal{F})$ with $\emptyset \in \mathcal{F}$ and closure operator σ on $\mathcal{F}$
Output: family of closed sets $\sigma(\mathcal{F})$

MAIN:

1. **print** $\sigma(\emptyset)$
2. LIST $(\sigma(\emptyset), \emptyset)$

LIST(C, B):

1. choose an element $e \in E \setminus (C \cup B)$ satisfying $C \cup \{e\} \in \mathcal{F}$ if such an e exists; otherwise **return**
2. $C' \leftarrow \sigma(C \cup \{e\})$
3. **if** $C' \cap B = \emptyset$ **then**
4. **print** C'
5. LIST (C', B)
6. LIST $(C, B \cup \{e\})$

In this section we analyze Algorithm 1—a simple divide-and-conquer algorithm solving Problem 1 that recursively applies the following principle: For the current closed set C, first list all closed supersets of C containing some augmentation element e and then all closed supersets of C not containing e. This is a well-known listing scheme (see for instance [3]). However, in contrast to other closed set listing algorithms, it is defined for any $\mathcal{F} \subseteq \mathcal{P}(E)$ with $\emptyset \in \mathcal{F}$. We will show that, for strongly accessible set systems, this algorithm efficiently solves LIST-CLOSED-SETS. In contrast, if the input set system is not strongly accessible, Algorithm 1 is not even "correct" for the identity map as closure operator. This statement builds on the following notion of *correctness*:

(i) Algorithm 1 *behaves correctly* on input $(E, \mathcal{F})$ and σ if it exactly and non-redundantly prints the elements of $\sigma(\mathcal{F})$ for *all correct implementations of line 1* in LIST, i.e., for all correct choices of augmentation elements.

(ii) Moreover, we say that Algorithm 1 is correct for a set system $(E, \mathcal{F})$ if for *all closure operators* σ on $\mathcal{F}$ it behaves correctly on input $(E, \mathcal{F})$ and σ.

The motivation for requiring correct behavior for all choices of augmentation elements (item (i) above) is that, in case there is only some sequences of choices leading to the correct output, it is unclear how to find such a sequence in general.

Theorem 1. *Let $(E, \mathcal{F})$ be a set system with $\emptyset \in \mathcal{F}$. Algorithm 1 is correct for $(E, \mathcal{F})$ if and only if $(E, \mathcal{F})$ is strongly accessible.*

PROOF. ("$\Leftarrow$") Let $(E, \mathcal{F})$ be strongly accessible and σ be a closure operator on $\mathcal{F}$. For $C, B \subseteq E$, let $\mathcal{C}(C, B) = \{C' \in \sigma(\mathcal{F}) \colon C' \supset C \wedge C' \cap B = \emptyset\}$. We prove by induction on the height h of the recursion tree of $\text{LIST}(C, B)$ that

$$\text{LIST}(C, B) \text{ prints exactly } \mathcal{C}(C, B) \text{ and} \tag{1}$$

$$\text{LIST}(C, B) \text{ prints no set more than once.} \tag{2}$$

Since MAIN calls $\text{LIST}(\sigma(\emptyset), \emptyset)$ and prints only the closed set $\sigma(\emptyset)$, this concludes the proof of the sufficiency. For $h = 0$, no augmentation element is selected in line 1. Therefore, $\text{LIST}(C, B)$ prints no element on the one hand and, as $(E, \mathcal{F})$ is strongly accessible, $\mathcal{C}(C, B) = \emptyset$ on the other hand, from which (1) and (2) directly follow. For the induction step $h > 0$ we must have that an augmentation element $e \in E \setminus (C \cup B)$ has been selected in line 1. We distinguish two cases depending on $C' = \sigma(C \cup \{e\})$ computed in line 2:

(i) Suppose $C' \cap B \neq \emptyset$. Then the set of closed sets printed by $\text{LIST}(C, B)$ is equal to the set $\mathcal{L}$ printed by $\text{LIST}(C, B \cup \{e\})$. Applying the induction hypothesis to $\text{LIST}(C, B \cup \{e\})$, we get $\mathcal{L} = \mathcal{C}(C, B \cup \{e\})$ and (2). Thus, to prove (1) it suffices to show that $\mathcal{C}(C, B \cup \{e\}) = \mathcal{C}(C, B)$. Clearly, $\mathcal{C}(C, B \cup \{e\}) \subseteq \mathcal{C}(C, B)$. Conversely, let $C'' \in \mathcal{C}(C, B)$. Then $e \notin C''$, as otherwise we would have $\sigma(C \cup \{e\}) \subseteq \sigma(C'') = C''$ by the monotonicity and idempotence of σ and hence, $C'' \cap B \neq \emptyset$ contradicting $C'' \in \mathcal{C}(C, B)$. Thus, $C'' \in \mathcal{C}(C, B \cup \{e\})$.

(ii) Suppose $C' \cap B = \emptyset$. Then the family printed by $\text{LIST}(C, B)$ is equal to $\{C'\} \cup \mathcal{L}_1 \cup \mathcal{L}_2$, where C' is the closed set printed in line 4 and $\mathcal{L}_1, \mathcal{L}_2$ are the families printed by $\text{LIST}(C', B)$ and $\text{LIST}(C, B \cup \{e\})$, respectively. Let $C'' \in \mathcal{C}(C, B)$ with $e \in C''$ and $C' \neq C''$. Then $C'' \in \mathcal{C}(C', B)$ for $C \cup \{e\} \subseteq C''$ and $C' = \sigma(C \cup \{e\}) \subset \sigma(C'') = C''$. Thus, $\mathcal{C}(C, B) = \{C'\} \cup \mathcal{C}(C', B) \cup \mathcal{C}(C, B \cup \{e\})$, from which (1) directly follows by applying the induction hypothesis to $\text{LIST}(C', B)$ and $\text{LIST}(C, B \cup \{e\})$. Since $C' \notin \mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$ because $\mathcal{C}(C', B) \cap \mathcal{C}(C, B \cup \{e\}) = \emptyset$, we get (2) by applying the induction hypothesis to $\text{LIST}(C', B)$ and $\text{LIST}(C, B \cup \{e\})$.

("$\Rightarrow$") Suppose that $(E, \mathcal{F})$ is not strongly accessible. Then choose $X = \{x_1, \ldots, x_k\} \in \mathcal{F}$ minimal such that there is a $Y \in \mathcal{F}$ with $X \subset Y$ and $(X \cup \{y\}) \notin \mathcal{F}$ for all $y \in Y \setminus X$. Let σ be the identity map on $\mathcal{F}$ that is clearly a closure operator. We show that there are possible choices of augmentation elements that result in an incorrect output. Consider the sequence of recursive calls

$$\text{LIST}(X_0, \emptyset), \text{LIST}(X_1, \emptyset), \ldots, \text{LIST}(X_k, \emptyset)$$

with $X_i = \{x_1, \ldots, x_i\}$, which arises as prefix of the call sequence in case x_i is chosen as augmentation element in line 1 of $\text{LIST}(X_{i-1}, \emptyset)$. If there is an augmentation element e in $\text{LIST}(X_k, \emptyset)$ then $e \notin Y$ by the choice of X and Y.

Thus, Y can neither be found in that incarnation of LIST nor in any of its subcalls LIST(C, B) because $e \in C$ for all such calls. For all other subsequent calls backtracking has occurred at least once. Thus, $x_i \in B$ for some $i \in \{1, \ldots, k\}$ for all such calls LIST(C, B), and consequently Y will be rejected by the check in line 3 in case it is found. Altogether, Algorithm 1 does not print Y and, hence, is incorrect for input $(E, \mathcal{F})$ and σ. □

As a byproduct of Theorem 1 we get an *algorithmic characterization* of strong accessibility. Note that the degree of freedom of the closure operator is not necessary for the proof of the "only if"-part of that theorem. Thus, the characterization can simply be stated as in Theorem 2 below. Recall that the underlying notion of correctness involves correct computation for *all* implementations that are in accord with the pseudo-code specification of Algorithm 1; in particular this means correctness for all valid choices of augmentation elements in line 1.

Theorem 2. *Let $(E, \mathcal{F})$ be a set system with $\emptyset \in \mathcal{F}$. Then $(E, \mathcal{F})$ is strongly accessible if and only if Algorithm 1 correctly lists $\mathcal{F}$ on input $(E, \mathcal{F})$ and the identity operator on $\mathcal{F}$ (as closure operator).*

This characterization is a distinctive feature that separates Algorithm 1 from related algorithms discussed in Section 8.

4. Performance

Algorithm 2 Modified Algorithm

MAIN′:

```
1.  initialize C and B to be empty stacks
2.  let e_1, ..., e_n be an arbitrary fixed ordering of E
3.  push all c ∈ σ(∅) onto C
4.  print C
5.  LIST′(1)
```

LIST′(d):

```
1.  push ⊥ onto C, push ⊥ onto B //set restoration point
2.  for i = 1, ..., n do
3.    if e_i ∈ (C ∪ B) or C ∪ {e_i} ∉ F then continue with next i
4.    push all c ∈ (σ(C ∪ {e_i}) \ C) onto C
5.    if C ∩ B = ∅ then
6.      if d is even then
7.        print C
8.        LIST′(d + 1)
9.      else
10.       LIST′(d + 1)
11.       print C
12.   while top of C not equal to ⊥ do pop C //restore C
13.   push e_i onto B
14. while top of B not equal to ⊥ do pop B //restore B
15. pop C, pop B //remove restoration point
```

We now turn to the complexity of Problem 1 restricted to strongly accessible set systems. In addition to the size of the input ground set E, the complexity also depends on the representation of the set system and on the closure operator. Accordingly, we will study the time and space complexity also in terms of those of (i) checking membership in $\mathcal{F}$ and (ii) computing the closure of an element in $\mathcal{F}$. We denote by $T_{\mathcal{F}}$, $S_{\mathcal{F}}$, T_σ, and S_σ the maximum time and space requirements of these operations for an input of size $|E|$, respectively. We assume that single elements of E can be stored, compared, or otherwise manipulated in time and space $O(1)$. For environments violating this assumption all complexities have to be multiplied by $\log |E|$.

For our analysis we consider a modified formulation of the divide-and-conquer algorithm given in Algorithm 2. While the two algorithms are input-output equivalent, Algorithm 2 allows us to prove a stronger performance statement. The changes are:

M1 The tail-recursion of the LIST procedure is replaced by a for-loop iterating over all potential augmentation elements $e \in E \setminus (C \cup B)$ using an arbitrary but fixed ordering of E.

M2 The parameters of the LIST procedure are replaced by global variables. In particular this can be realized by implementing them as stacks of single elements of the ground set because—due to the recursive structure of the algorithm—elements are added and removed in a last-in-first-out fashion.

M3 For odd recursion depths the order of lines 4 and 5 of Algorithm 1 is changed, i.e., the new closed set is printed only after the recursive call backtracked.

Modifications M1 and M2 are only equivalent reformulations. The third modification follows an idea described by Nakano and Uno [7] in the context of listing trees of a fixed diameter. It does not change what is printed—and thus does not affect correctness—but "holds back" some of the output elements for a certain time until eventually printing them. As a result the moments in which closed sets are printed are more evenly distributed during the running time of the algorithm. This improves the *delay*, i.e., the maximum time between the generation of two consecutive closed sets, by a factor of $|E|$ over that of the original formulation, without changing the total time.

Theorem 3. *Restricted to strongly accessible set systems,* LIST-CLOSED-SETS *can be solved with*

$$\textit{delay } O(|E|\ (T_{\mathcal{F}} + T_\sigma + |E|)) \quad \textit{, and} \tag{3}$$

$$\textit{space } O(|E| + S_{\mathcal{F}} + S_\sigma) \quad . \tag{4}$$

PROOF. Let $n = |E|$. To see the *delay*, first observe that the algorithm cannot backtrack more than two times without printing a new set or terminating. Moreover, if LIST$'(d)$ is called with an even d, without printing a new closed set or backtracking there can be at most

- n membership checks (cost $T_{\mathcal{F}}$ each),
- n closure computations (cost T_σ each) and

- n manipulations and accesses to a constant number of variables of size at most $O(n)$ (cost $O(n)$ each).

Finally, for LIST$'(d)$ with an odd d there cannot be more than the same operations of time $O(T_{\mathcal{F}} + T_\sigma + n)$ without calling LIST$'$ with an even d or backtracking. The claimed delay follows by noting that MAIN$'$ prints a set and calls LIST$'$ after time $O(T_\sigma + n)$.

The *space complexity* is straightforward: since $C \cup B \subseteq E$ always holds, there are never more than $O(n)$ elements to be stored. Equation (4) directly follows. □

From this theorem it immediately follows a bound on the *total running time* (i.e., $|\sigma(\mathcal{F})|$ times the delay given in Equation (3)) because of the exactness and non-redundancy guaranteed by Theorem 1. In fact this bound on the total time and the space bound can already be shown for the algorithm incorporating only the equivalent reformulations M1 and M2 above. The amortized cost of a single invocation of LIST is similar to the one found in the proof of Theorem 3 and non-redundancy and exactness imply that LIST(C', B) in line 5 of Algorithm 1 is called at most once for each $C' \in \sigma(\mathcal{F})$. Closing this section, we can thus note:

Remark 1. *Restricted to strongly accessible set systems, Algorithm 1 can be implemented to solve* LIST-CLOSED-SETS *with total time*

$$O(|E|\ (T_{\mathcal{F}} + T_\sigma + |E|)\ |\sigma(\mathcal{F})|)$$

and space as given in Equation (4)*.*

5. Problem Complexity for Accessible Set Systems

Clearly, for any set system $(E, \mathcal{F})$ and closure operator σ on $\mathcal{F}$, $\sigma(\mathcal{F})$ can be listed in total time $O(2^n)$ by a deterministic algorithm that has access to $\mathcal{F}$ only by means of membership oracle and closure computations, if the invocation of the membership oracle and the closure computation are both charged by unit time. Theorem 4 below not only shows that this bound cannot be substantially improved for accessible set systems, but also implies that there is no deterministic algorithm solving LIST-CLOSED-SETS for this problem fragment in output polynomial time, i.e., by an algorithm having a time complexity that is polynomially bounded in $n + |\sigma(\mathcal{F})|$.

Theorem 4. *For accessible set systems $(E, \mathcal{F})$ and closure operators σ on $\mathcal{F}$ such that $|\sigma(\mathcal{F})| \leq 2$, there is no deterministic algorithm that has access to $\mathcal{F}$ only by means of membership oracle and closure computations, and correctly solves problem* LIST-CLOSED-SETS *by invoking the membership oracle and computing the closure operator at most $2^{n/4}$ times where $n = |E|$.*

PROOF. Let $\mathcal{A}$ be a deterministic algorithm solving the problem described in the claim by invoking the membership oracle and computing the closure operator at most $2^{n/4}$ times. We show that $\mathcal{A}$ is incorrect by constructing two problem instances such that $\mathcal{A}$ fails to compute the correct closed set family for at least one of them.

For a positive integer $n > 4$ dividable by 4, consider the set system $(E, \mathcal{F})$ with $E = X \cup Y$ such that X, Y are disjoint sets, both of cardinality $n/2$, $\mathcal{F} = \mathcal{P}(X)$, and define the function $\sigma : \mathcal{F} \to \mathcal{F}$ by $\sigma(F) = X$ for all $F \in \mathcal{F}$. Clearly $(E, \mathcal{F})$ is accessible, σ is a closure operator, and $|\sigma(\mathcal{F})| = 1$. Thus, $(E, \mathcal{F})$ and σ form an instance of the problem described in the claim.

Since $\mathcal{A}$ invokes the membership oracle or computes the closure operator at most $2^{n/4} < \binom{n/2}{n/4}$, there is at least one subset of Y of cardinality $n/4$, say $Y' = \{e_1, \ldots, e_{n/4}\}$, such that $\mathcal{A}$ on input $\mathcal{F}$ and σ does not access (neither by the membership oracle nor by the closure computation) $Y' \cup F$ for all $F \subseteq X$. For the same reason there is a subset $X' \subseteq X$ of cardinality $n/4$ such that $\mathcal{A}$ does not access $X' \cup F$ for all $F \subseteq Y$. Let $X \setminus X' = \{e_{n/4+1}, \ldots, e_{n/2}\}$ and consider the set system $(E, \mathcal{F}')$ with $\mathcal{F}' = \mathcal{F} \cup \{S_1, \ldots, S_{n/2}\}$, where $S_0 = X'$ and $S_i = S_{i-1} \cup \{e_i\}$ for every $i = 1, \ldots, n/2$. Note that $S_{n/2} = X \cup Y'$. Let the function $\sigma' : \mathcal{F}' \to \mathcal{F}'$ be defined by

$$\sigma' : F \mapsto \begin{cases} X, \text{ if } F \in \mathcal{F} \\ S_{n/2}, \text{ otherwise } . \end{cases}$$

One can easily check that $(E, \mathcal{F}')$ is accessible, σ' is a closure operator on $\mathcal{F}'$, and $|\sigma'(\mathcal{F}')| = 2$. Hence, $(E, \mathcal{F}')$ and σ' form a second instance of the problem defined in the statement.

Let $A_1(F_1), \ldots, A_k(F_k)$ be the sequence of membership queries and closure computations performed by $\mathcal{A}$ on the first instance defined by $(E, \mathcal{F})$ and σ. That is, $A_i(F_i)$ is either $M_{\mathcal{F}}(F_i)$ or $\sigma(F_i)$ for some $F_i \in \mathcal{P}(E) \setminus \{S_0, S_1, \ldots, S_{n/2}\}$ for every $i = 1, \ldots, k$, where $M_{\mathcal{F}}$ denotes the membership oracle for $\mathcal{F}$. Since $\mathcal{A}$ is deterministic, $F_i \notin \{S_0, S_1, \ldots, S_{n/2}\}$, and $M_{\mathcal{F}}(X) = M_{\mathcal{F}'}(X)$ and $\sigma(X) = \sigma'(X)$ for every $X \in \mathcal{P}(E) \setminus \{S_0, S_1, \ldots, S_{n/2}\}$, $\mathcal{A}$ will perform the same sequence of membership queries and closure computations for the second instance defined by $(E, \mathcal{F}')$ and σ' and generate the same family of closed sets. But this implies that $\mathcal{A}$ is incorrect on at least one of the two instances, as $\sigma(\mathcal{F}) \neq \sigma'(\mathcal{F}')$. □

6. Support-Closed Sets

So far we have defined a closed set as a fixpoint of some closure operator. In data mining a different notion of closedness is used. To define it, we first recall some necessary definitions from frequent pattern mining. A *dataset* over a set E is a multiset $\mathcal{D}$ of subsets of E. The elements of $\mathcal{D}$ are called *transactions*. We say that $\mathcal{D}$ is *non-redundant* if for all $e \in E$ there is a $D \in \mathcal{D}$ with $e \notin D$, i.e., there is no element that is contained in every transaction. For a set $X \subseteq E$, the *support set* of X with respect to $\mathcal{D}$, denoted $\mathcal{D}[X]$, is the multiset of transactions of $\mathcal{D}$ containing X.

Based on support sets one can define the following notion of closedness: a set $X \in \mathcal{F}$ is *support-closed* if $X \subset Y$ implies $\mathcal{D}[X] \supset \mathcal{D}[Y]$ for every $Y \in \mathcal{F}$. By $\mathcal{SC}(\mathcal{F}, \mathcal{D})$ we denote the family of all support-closed sets in $\mathcal{F}$ with respect to $\mathcal{D}$. Note that for $\emptyset \in \mathcal{F}$ it holds that, for all $\mathcal{D}$, $\mathcal{D}$ is non-redundant if and only if $\emptyset \in \mathcal{SC}(\mathcal{F}, \mathcal{D})$. We include this requirement in our formal problem statement for listing closed sets. Though it is a minor restriction, it makes our results applicable to more problems of practical interest (for instance Problem 5 from Section 7).

Problem 2 (LIST-SC-SETS). *Given* a set system $(E, \mathcal{F})$ and a non-redundant dataset $\mathcal{D}$ over E, *list* the family of support-closed sets $\mathcal{SC}(\mathcal{F}, \mathcal{D})$.

The two notions of closedness, based on support sets and based on closure operators, are not equivalent: there are set systems and datasets such that no closure operator exists having exactly the support-closed sets as fixpoints. Hence Algorithm 1 is not generally applicable to Problem 2. Indeed, even when restricted to independence systems, LIST-SC-SETS is intractable (see Example 1).

Theorem 5. *There is no algorithm solving* LIST-SC-SETS *restricted to independence systems in output polynomial time (unless* ***P=NP****).*

PROOF. Let $(E, \mathcal{F})$ be an independence system. For $\mathcal{D} = \{\emptyset, E\}$ the problem of listing $\mathcal{SC}(\mathcal{F}, \mathcal{D})$ is equivalent to listing the *bases* of $(E, \mathcal{F})$, i.e., the maximal elements of $\mathcal{F}$. For the latter problem it was shown by Lawler et al. [10] that it cannot be solved in output polynomial time (unless **P=NP**). □

If, however, such a closure operator exists, we call it *support closure operator* of $\mathcal{F}$ with respect to $\mathcal{D}$. In case of existence such an operator is uniquely defined as follows:

Lemma 6. *Let $(E, \mathcal{F})$ be a set system and $\mathcal{D}$ a dataset over E. If a support-closure operator σ on $\mathcal{F}$ with respect to $\mathcal{D}$ exists then it is well-defined by*

$$\begin{aligned} \sigma(F) &= \max \Sigma(F) \\ \text{where } \Sigma(F) &= \{F' \in \mathcal{F} \colon F \subseteq F' \wedge \mathcal{D}[F] = \mathcal{D}[F']\} \ , \end{aligned} \tag{5}$$

i.e., the family $\Sigma(F)$ has a unique maximal element for all $F \in \mathcal{F}$.

PROOF. Note that for all $F \in \mathcal{F}$,

$$F \in \mathcal{SC}(\mathcal{F}, D) \iff \exists G \in \mathcal{F} \text{ such that } F \text{ is maximal in } \Sigma(G) \ . \tag{6}$$

Let σ be a support closure operator on $\mathcal{F}$ and F' be a maximal element in $\Sigma(F)$. Then F' is support-closed by (6) and hence $\sigma(F') = F'$. Since $F \subseteq F'$, we have $F \subseteq \sigma(F) \subseteq \sigma(F') = F'$ by extensivity and monotonicity of σ. But this implies $\mathcal{D}[F] = \mathcal{D}[\sigma(F)]$, as $\mathcal{D}[F] = \mathcal{D}[F']$ by $F' \in \Sigma(F)$. Thus $\sigma(F) \in \Sigma(F)$. But then, $\sigma(F)$ must be maximal in $\Sigma(F)$, as it is maximal in $\Sigma(\sigma(F))$ by (6). Hence, $\sigma(F) = F'$ because $\sigma(F) \subseteq F'$. □

A unique maximal element of $\Sigma(F)$ does not always exist (e.g., $F = \emptyset$ with $\mathcal{F} = \{\emptyset, \{a\}, \{b\}\}$ with $\mathcal{D} = \{\{a, b\}\}$), and even if it exists, σ defined by (5) is not always monotone (consider, e.g., $\mathcal{F} = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}\}$ with $\mathcal{D} = \{\{a, b\}, \{a, b, c\}\}$). If, however, $\max \Sigma(F)$ is unique for every $F \in \mathcal{F}$ and σ as defined above is monotone, the reverse of the above lemma holds.

We now show that confluence characterizes the existence of the support closure operator for arbitrary non-redundant datasets. That is, restricted to confluent set systems, Problem 2 is a subproblem of Problem 1.

Theorem 7. *Let $(E, \mathcal{F})$ be a set system. The support closure operator for $\mathcal{F}$ with respect to $\mathcal{D}$ exists for all non-redundant datasets $\mathcal{D}$ over E if and only if $(E, \mathcal{F})$ is confluent.*

PROOF. ("⇐") Suppose $(E, \mathcal{F})$ is confluent and let $\mathcal{D}$ be a non-redundant dataset over E. Let $\sigma(F)$ be $\max \Sigma(F)$ for all $F \in \mathcal{F}$ as defined in (5). We prove that σ is a support-closure operator by showing that

(i) σ is a function,

(ii) $\sigma(F)$ is support closed for all $F \in \mathcal{F}$, and

(iii) σ is a closure operator.

Regarding (i), we show that for all $F \in \mathcal{F}$, there exists a unique maximal element in $\Sigma(F)$. For $F = \emptyset$ this is trivial because $\Sigma(\emptyset) = \{\emptyset\}$ by the non-redundancy of $\mathcal{D}$. Let $F \neq \emptyset$. Existence is implied by the finiteness of $\mathcal{F}$. For uniqueness assume there are distinct sets $F', F'' \in \mathcal{F}$ that are both maximal in $\Sigma(F)$. Since $F \neq \emptyset$, $F \subseteq F'$, and $F \subseteq F''$, it follows from the confluence of $\mathcal{F}$ that $(F' \cup F'') \in \mathcal{F}$. As $\mathcal{D}[F' \cup F''] = \mathcal{D}[F'] \cap \mathcal{D}[F'']$ and $\mathcal{D}[F'] = \mathcal{D}[F''] = \mathcal{D}[F]$, we have $\mathcal{D}[F' \cup F''] = \mathcal{D}[F]$. Thus $F' \cup F'' \in \Sigma(F)$ contradicting the maximality of F' and F''. Hence, there is a unique maximal element of $\Sigma(F)$.

Property (ii) is immediate by (6), as $\sigma(F)$ is maximal in $\Sigma(F)$. To see (iii), the extensivity follows by definition. For idempotence we have $\sigma(F) = \max \Sigma(F) \in \Sigma(F)$ implying $\mathcal{D}[\sigma(F)] = \mathcal{D}[F]$. Hence $\sigma(\sigma(F)) = \sigma(F)$, as $\sigma(F)$ is maximal. For monotonicity, let $F', F'' \in \mathcal{F}$ with $F' \subseteq F''$. The case $F' = \emptyset$ is trivial because $\sigma(\emptyset) = \emptyset$, as $\mathcal{D}$ is non-redundant. Let $F' \neq \emptyset$. Since $F' \subseteq \sigma(F')$ and $F' \subseteq F''$, $\sigma(F') \cup F'' \in \mathcal{F}$ by the confluence of $\mathcal{F}$. For the support of $\sigma(F') \cup F''$ we have $\mathcal{D}[\sigma(F') \cup F''] = \mathcal{D}[\sigma(F')] \cap \mathcal{D}[F'']$, which, in turn, is equal to $\mathcal{D}[F'']$ because $\mathcal{D}[\sigma(F')] = \mathcal{D}[F']$ by the definition of σ and $\mathcal{D}[F''] \subseteq \mathcal{D}[F']$ by $F' \subseteq F''$. Hence $\sigma(F') \cup F'' \in \Sigma(F'')$ and $\sigma(F') \subseteq \sigma(F') \cup F'' \subseteq \sigma(F'')$ by maximality of $\sigma(F'')$.

("⇒") Suppose that for all non-redundant datasets $\mathcal{D}$ over E the support closure of $\mathcal{F}$ with respect to $\mathcal{D}$ exists. In order to show that $(E, \mathcal{F})$ is confluent let $I, X, Y \in \mathcal{F}$ with $I \neq \emptyset$, $I \subseteq X$, and $I \subseteq Y$. We show that $\sigma(I) = X \cup Y$ for the support closure operator $\sigma \colon \mathcal{F} \to \mathcal{F}$ with respect to the dataset $\mathcal{D} = \{\emptyset, X \cup Y\}$. Since, on the one hand, σ is support preserving it follows that $\sigma(I) \subseteq X \cup Y$. On the other hand, $\sigma(I)$ is support-closed. Together with $\mathcal{D}[I] = \mathcal{D}[X] = \mathcal{D}[Y]$ this implies that $X \subseteq \sigma(I)$ and $Y \subseteq \sigma(I)$. Hence, it also holds that $\sigma(I) \supseteq X \cup Y$ as required. □

Theorem 7 can be used to characterize the instances of LIST-SC-SETS that are also instances of LIST-CLOSED-SETS. But even in case that the support closure operator exists, it is unclear whether its computation is tractable. In the following lemma we show that if a support closure operator has a strongly accessible domain, it can be computed efficiently by reducing it to the *augmentation problem* (line 1 of Algorithm 1), i.e., the problem to find an element $e \in E \setminus (B \cup C)$ with $(C \cup \{e\}) \in \mathcal{F}$ or decide that none exists, given $B, C \subseteq E$. We denote the required time to solve this problem by T_a. Note that it can always be solved with $|E \setminus (C \cup B)|$ membership queries (and no additional space). We nevertheless make T_a an explicit parameter of the result below, because usually it can be implemented more efficiently than by the naive approach via membership queries (see the examples from Section 7).

Lemma 8. *Let $(E, \mathcal{F})$ be a strongly accessible set system and $\mathcal{D}$ be a non-redundant dataset over E. If the support closure operator of $\mathcal{F}$ with respect to $\mathcal{D}$ exists it can be computed in time $O(|E|\ (|\mathcal{D}| + T_a))$ and space $S_{\mathcal{F}}$.*

PROOF. Let σ be a support closure operator. Define $F_0 = F$ and

$$F_{i+1} = \begin{cases} F_i \cup \{e\} & \text{if } \exists e \in \bigcap \mathcal{D}[F] \setminus F_i \text{ such that } F_i \cup \{e\} \in \mathcal{F} \\ F_i & \text{otherwise} \end{cases}$$

for $i \geq 0$. Since the sequence $F = F_0 \subseteq F_1 \subseteq \ldots$ is bounded by $\bigcap \mathcal{D}[F]$, there is a smallest index $k < |E|$ such that $F_k = F_{k+1}$. Clearly, $\mathcal{D}[F] = \mathcal{D}[F_i]$ and hence, $F_i \in \Sigma(F)$ for every $i = 0, 1, \ldots, k$. Since there is no further augmentation element $e \in (\bigcap \mathcal{D}[F] \setminus F_k)$ and $(E, \mathcal{F})$ is strongly accessible, it follows that F_k is maximal in $\Sigma(F)$. Thus, by Lemma 6, $F_k = \sigma(F)$ as required. By the definition above, F_k can be computed by calculating $\bigcap \mathcal{D}[F]$ and by finding at most $|E|$ augmentation elements. The statement about the time then follows because $\bigcap \mathcal{D}[F]$ can be computed in time $O(|E|\ |D|)$. For the required space note that for the computation of the result F_k there is no additional storage required beside that for computing an augmentation element, which can be reused. □

Combining Theorem 3 with the results of this section, we can identify a fairly general, tractable subproblem of LIST-SC-SETS. While the theorem below may not yield the strictest bounds for concrete problems where more structural assumptions hold, its conditions can usually be checked easily and it serves as a baseline for more specialized methods.

Theorem 9. *Restricted to set systems that are confluent and strongly accessible* LIST-SC-SETS *can be solved with delay* $O\Big(|E|^2\ (|\mathcal{D}| + T_a)\Big)$ *respectively delay* $O\Big(|E|^2\ (|\mathcal{D}| + |E|\, T_{\mathcal{F}})\Big)$ *and space* $O(|E| + S_{\mathcal{F}})$.

Note that it is crucial for Theorem 9 that Theorem 3 holds for closure operators that are only a partial function of the power set of the ground set. The support closure operator is in general not defined for arbitrary members of the power set.

7. Applications

In this section we present three listing problems motivated by data mining applications and show that they can be solved with polynomial delay and space. We derive all these positive results by applying Theorems 3 and 9.

Frequent Sets. As a first example, consider the data mining problem of listing all support-closed frequent sets (see, e.g., [4, 5]). For an integer *frequency threshold* $t > 0$, a subset $X \subseteq E$ is *t-frequent* if $|\mathcal{D}[X]| \geq t$.

Problem 3 (LIST-SC-FREQUENT-SETS**).** *Given* a dataset $\mathcal{D}$ over a finite set E and an integer frequency threshold $t > 0$, *list* all subsets of E that are t-frequent and support-closed with respect to $\mathcal{D}$.

Clearly, the family of t-frequent sets always forms an independence system. If we discard the frequency requirement the underlying set system is $\mathcal{F} = \mathcal{P}(E)$, for which the support closure operator always exists and is defined by $\sigma(F) = \bigcap \mathcal{D}[F]$. It can be computed in time $O(\|\mathcal{D}\|)$, where $\|\mathcal{D}\| = \sum_{D \in \mathcal{D}} |D|$ denotes the size of $\mathcal{D}$. Notice that if we restrict σ to the family $\mathcal{F}_t = \{F \subseteq E\colon$

$|\mathcal{D}[F]| \geq t\}$ of t-frequent sets the resulting mapping $\sigma_t \colon \mathcal{F}_t \to \mathcal{F}_t$ is the support closure operator of $(E, \mathcal{F}_t)$ with respect to $\mathcal{D}$. As $\mathcal{F}_t$ is an independence system and a membership test can be performed in time $O(\|\mathcal{D}\|)$, by Theorem 3 we get:

Corollary 10. *The* LIST-SC-FREQUENT-SETS *problem can be solved with delay* $O(|E|\ \|\mathcal{D}\|)$*, and space* $O(|E|)$*.*

In this example we used a general observation about support closure operators of a set system $(E, \mathcal{F})$: their restriction to the family $\mathcal{F}_t$ of t-frequent sets of $\mathcal{F}$ remains a support closure operator (with respect to the same dataset $\mathcal{D}$). Thus, if $\mathcal{F}$ is confluent and strongly accessible, Theorem 9 can still be applied when a frequency constraint is added to Problem 2. Though Theorem 9 could also be used to get a positive result on LIST-SC-FREQUENT-SETS, we applied Theorem 3 because the support closure operator for frequent sets can be computed faster than by the algorithm used in the proof of Lemma 8. The bound in Corollary 10 is actually equal to the best known theoretical bound for LIST-SC-FREQUENT-SETS achieved by the LCM-algorithm [5].

Poset Ideals. The next example makes use of the fact that greedoids are strongly accessible. Let $(E, \leq)$ be a poset. Then $F \subseteq E$ is called a (lower) *ideal* if for all $e \in F$ and for all $e' \in E$, $e' \leq e$ implies $e' \in F$. Using this notion, we can state the following listing problem:

Problem 4 (LIST-SC-IDEALS**).** *Given* a finite poset $(E, \leq)$ and a non-redundant dataset $\mathcal{D}$ over E, *list* the family of ideals of $(E, \leq)$ that are support-closed with respect to $\mathcal{D}$.

This problem is motivated by *recommendation systems* where partial orders are used for modeling a collection of user preferences (see, e.g., [11]). We show that it can be solved with polynomial delay and space. Let $\mathcal{F}$ be the family of ideals of $(E, \leq)$. Then $(E, \mathcal{F})$ forms a greedoid, the so-called *poset greedoid* [12], which implies that $(E, \mathcal{F})$ is strongly accessible. Furthermore, $(E, \mathcal{F})$ is confluent as poset greedoids are closed under union. An augmentation element (see line 1 of Algorithm 1) can be found in time $O(\|{\leq}\|)$ by touching each element (x, y) of the direct successor relation of $\leq$ and checking whether $x \in F$ and $y \notin (F \cup B)$. Altogether, by Theorem 9, we have the following result:

Corollary 11. *The* LIST-SC-IDEALS *problem can be solved with delay* $O\Big(|E|^2\ (|\mathcal{D}| + \|{\leq}\|)\Big)$ *and space* $O(|E|)$*.*

Connected Induced Subgraphs. Finally, we consider the problem of listing all connected induced subgraphs of a graph $G = (V, E)$ that are support-closed with respect to a dataset over V. Such datasets can for instance model movements of individuals in a street network and occur in *track mining* applications (see, e.g., [13]). The formal problem statement is:

Problem 5 (LIST-SC-CONNECTED-VERTICES**).** *Given* an undirected graph $G = (V, E)$ and a non-redundant dataset $\mathcal{D}$ over V, *list* the family of sets $F \subseteq V$ inducing connected subgraphs[1] of G that are support-closed with respect to $\mathcal{D}$.

[1]Note that in contrast to standard problems in graph mining, Problem 5 does not rely by any means on subgraph isomorphism. In this article, support-closedness is always only defined with respect to set inclusion.

For a graph $G = (V, E)$ and $X \subseteq V$ let $G[X]$ denote the subgraph of G induced by X. We note that the family of vertex sets that induce connected subgraphs of G is not an independence system because a subgraph of a connected graph is not necessarily connected. It is, however, strongly accessible and also confluent.

Lemma 12. *For a graph $G = (V, E)$ let $\mathcal{F} = \{X \subseteq V : G[X]$ is connected$\}$. Then $(V, \mathcal{F})$ is strongly accessible and confluent.*

Proof. The confluence follows since the union of any two connected subgraphs of G with non-disjoint vertex sets is also connected. For strong accessibility, let $X, Y \in \mathcal{F}$ with $X \subset Y$. Assume there is no vertex $v \in Y \setminus X$ such that $X \cup \{v\} \in \mathcal{F}$. Then X and $Y \setminus X$ are disconnected in $G[Y]$ contradicting the choice of Y. □

As for the previous example, an augmentation element can be found in time $O(|E|)$ by touching each edge e once and checking whether $x \in F$ and $y \notin (F \cup B)$ for $x, y \in e$. Therefore we obtain by Theorem 9:

Corollary 13. *The* list-sc-connected-vertices *problem can be solved with delay $O\Big(|V|^2\ (|\mathcal{D}| + |E|)\Big)$ and space $O(|V|)$.*

Note that it is crucial for the list-sc-connected-vertices problem that the datasets are non-redundant. Otherwise the support closure operator does not always exist as the following example shows:

Example 1. Let $G = (\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$ a path of length two and $\mathcal{D} = \{\{a, c\}\}$. Then there is no support closure operator σ for $\mathcal{F}$ with respect to $\mathcal{D}$ because each possible choice of $\sigma(\emptyset)$, either $\{a\}$ or $\{c\}$, would violate monotonicity.

8. Discussion

The previous section has demonstrated that it is useful to have a listing algorithm for closure operators that are only defined on a subset of the power set $\mathcal{P}(E)$; particularly for subsets that are not necessarily independence systems. Indeed, for the problems list-sc-ideals and list-sc-connected-vertices the support closure operator is in general only a partial function on $\mathcal{P}(E)$ because σ is undefined for non-ideals and for vertex sets inducing disconnected graphs, respectively. Moreover, the set systems considered for both problems are not necessarily independence systems. As a further remark, note that, for list-sc-ideals, the induced family of closed sets always forms a closure system because E is an ideal of $(E, \leq)$ and $\mathcal{F}$ is closed under intersection. In contrast, this is not the case for list-sc-connected-vertices.

In contrast to traditional algorithms that assume $\mathcal{F} = \mathcal{P}(E)$ (e.g., [1]), our former algorithm [8] as well as the one of Arimura and Uno [9] are also applicable to partially defined closure operators. In fact, as we show below, the class of set systems for which these algorithms are correct for *all* closure operators properly contains the class of strongly accessible set systems while it is properly contained in the class of accessible set systems. Thus, on the one hand, these algorithms are correct for a larger set of inputs than Algorithm 1, but, on the other hand, for non-strongly accessible set systems it is intractable to decide whether their

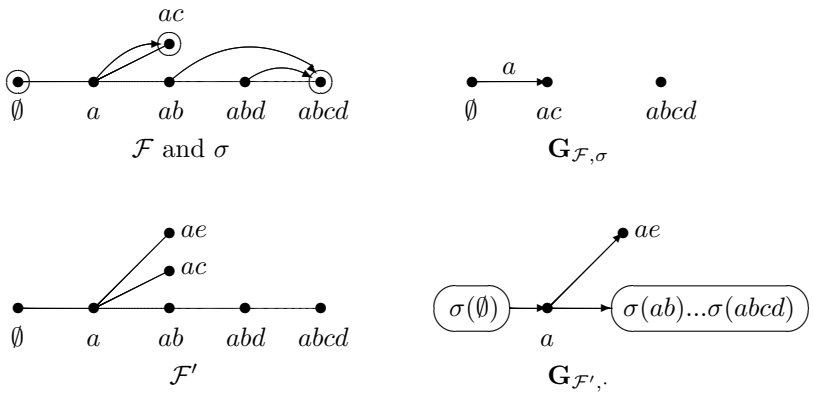

Figure 2: The accessible set system $(E, \mathcal{F})$ and the closure operator σ from Example 2 (top left) together with the corresponding generator graph $\mathbf{G}_{\mathcal{F}',\sigma}$ (top right). The closed set *abcd* is not reachable from $\emptyset$ in $\mathbf{G}_{\mathcal{F}',\sigma}$. In addition the accessible set system $(E', \mathcal{F}')$ from Example 3 (bottom left). The corresponding generator graphs are connected for all closure operators (sketch bottom right).

output is correct. In fact this observation sheds light on a distinctive feature of Algorithm 1: in contrast to the other algorithms it yields an exact algorithmic characterization of strong accessibility (see Section 3). In order to show the claims above, we briefly review both algorithms.

The first one [8] can be seen as a straightforward traversal, starting in $\sigma(\emptyset)$, of the *generator graph* $\mathbf{G}_{\mathcal{F},\sigma} = (\mathbf{V}, \mathbf{E})$ induced by the input set system $(E, \mathcal{F})$ and the closure operator σ, i.e., the directed graph with vertices $\mathbf{V} = \sigma(\mathcal{F})$ and edges

$$\mathbf{E} = \{(C, C') \in \sigma(\mathcal{F}) \times \sigma(\mathcal{F}) \colon \exists e \in E \setminus C,\, (C \cup \{e\}) \in \mathcal{F} \wedge \sigma(C \cup \{e\}) = C'\} \ .$$

Consequently, it runs in total time $O(Nn(T_{\mathcal{F}} + T_\sigma + n))$ but potentially exponential space $O(Nn)$, where $n = |E|$ and $N = |\sigma(\mathcal{F})|$, because it explicitly stores each visited vertex. The algorithm of Arimuro and Uno improves on the naive traversal of $\mathbf{G}_{\mathcal{F},\sigma}$ in that it traverses only a spanning tree without this explicit storage. This results in an efficient space complexity $O(n + S_{\mathcal{F}} + S_\sigma)$ and total time $O\big(N(n^3 T_{\mathcal{F}} + n^2 T_\sigma)\big)$.

However, both algorithms are incomplete, thus incorrect, if $\mathbf{G}_{\mathcal{F},\sigma}$ is unconnected and correct if it is connected. Recall that we call a closed set listing algorithm correct for a set system $(E, \mathcal{F})$ if it behaves correctly for *all* closure operators on $\mathcal{F}$. For the generator graph traversal algorithms this means that they are correct for $(E, \mathcal{F})$ if $\mathbf{G}_{\mathcal{F},\sigma}$ is connected for *all* closure operators σ on $\mathcal{F}$. It is straightforward to check that this connectivity criterion is, on the one hand, implied by $(E, \mathcal{F})$ being strongly accessible [8] and, on the other hand, at least requires $(E, \mathcal{F})$ to be accessible (e.g., choose σ to be the identity map). The two examples below (see also Figure 2) now show that this condition lies strictly between strong and ordinary accessibility.

First we give an accessible set system that is not strongly accessible and a closure operator such that the corresponding generator graph is unconnected.

Example 2. Let $(E, \mathcal{F})$ be the accessible set system defined by $E = \{a, b, c, d\}$ and $\mathcal{F} = \{\emptyset, a, ab, ac, abd, abcd\}$. Moreover, define $\sigma : \mathcal{F} \to \mathcal{F}$ by $\sigma(\emptyset) = \emptyset$, $\sigma(a) = \sigma(ac) = ac$, and $\sigma(ab) = \sigma(abd) = \sigma(abcd) = abcd$.

On the other hand, there are accessible set systems that are not strongly accessible and still have a connected generator graph for *all* closure operators. This is witnessed by the set system $(E', \mathcal{F}')$ of Example 3.

Example 3. Let $(E', \mathcal{F}')$ be the accessible set system defined by $E' = E \cup \{e\}$ and $\mathcal{F}' = \mathcal{F} \cup \{ae\}$ with $(E, \mathcal{F})$ of Example 2.

Although $(E', \mathcal{F}')$ just like $(E, \mathcal{F})$ is not strongly accessible, one can check that $\mathbf{G}_{\mathcal{F},\sigma}$ is connected for all closure operators σ on $\mathcal{F}$. This is caused by a being a maximal subset of two distinct globally maximal sets ae and $abcd$. It follows due to monotonicity that a is a fixpoint of all closure operators on $\mathcal{F}$. In the corresponding generator graph a will connect $\sigma(\emptyset)$ (possibly itself) to both "arms": $\{ae\}$ as well as $\{\sigma(ab), \sigma(ac), \sigma(abc), \sigma(abcd)\}$. Consequently and unlike Algorithm 1, the algorithms based on generator graph traversals are correct on $(E, \mathcal{F})$ for all closure operators.

Thus, these algorithms are correct for a larger set of inputs. In general, however, it is intractable to decide whether a given pair of an accessible set system and a closure operator induces a connected generator graph. This can be shown by using a similar construction as in the proof of Theorem 4. Accordingly, for a given accessible set system and closure operator it is in general intractable to decide whether the output of these algorithms is complete.

References

[1] B. Ganter, K. Reuter, Finding all closed sets : A general approach, Order 8 (1991) 283–290.

[2] B. Ganter, R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer Verlag, 1999.

[3] A. Gély, A generic algorithm for generating closed sets of a binary relation, in: Proceedings of ICFCA, Springer, 2005, pp. 223–234.

[4] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Efficient mining of association rules using closed itemset lattices, Inf. Syst. 24 (1) (1999) 25–46.

[5] T. Uno, T. Asai, Y. Uchida, H. Arimura, An efficient algorithm for enumerating closed patterns in transaction databases, in: Proceedings of Discovery Science, Springer, 2004, pp. 16–31.

[6] D. S. Johnson, C. H. Papadimitriou, M. Yannakakis, On generating all maximal independent sets, Inf. Process. Lett. 27 (3) (1988) 119–123.

[7] S.-I. Nakano, T. Uno, Constant time generation of trees with specified diameter, in: J. Hromkovic, M. Nagl, B. Westfechtel (Eds.), WG, Vol. 3353 of Lecture Notes in Computer Science, Springer, 2004, pp. 33–45.

[8] M. Boley, T. Horváth, A. Poigné, S. Wrobel, Efficient closed pattern mining in strongly accessible set systems (extended abstract), in: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007), Springer, 2007, pp. 382–389.

[9] H. Arimura, T. Uno, Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems, in: Proceedings of the SIAM International Conference on Data Mining (SDM 2009), SIAM, 2009, pp. 1087–1098.

[10] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Generating all maximal independent sets: Np-hardness and polynomial-time algorithms., SIAM J. Comput. 9 (3) (1980) 558–565.

[11] D. Bridge, A. Ferguson, Diverse product recommendations using an expressive language for case retrieval, Advances in Case-Based Reasoning (2002) 291–298.

[12] B. Korte, L. Lovász, Relations between subclasses of greedoids, Math. Meth. Oper. Res. 29 (7) (1985) 249–267.

[13] M. Nanni, B. Kuijpers, C. Körner, M. May, D. Pedreschi, Spatiotemporal data mining, in: Mobility, Data Mining and Privacy - Geographic Knowledge Discovery, 2008.