Annals of Mathematics and Artificial Intelligence manuscript No.
(will be inserted by the editor)

Fast Algorithms for Implication Bases and Attribute Exploration Using Proper Premises

Uwe Ryssel · Felix Distel · Daniel Borchmann.

Received: date / Accepted: date

Abstract A central task in formal concept analysis is the enumeration of a small base for the implications that hold in a formal context. The usual stem base algorithms have been proven to be costly in terms of runtime. Proper premises are an alternative to the stem base. We present a new algorithm for the fast computation of proper premises. It is based on a known link between proper premises and minimal hypergraph transversals. Two further improvements are made, which reduce the number of proper premises that are obtained multiple times and redundancies within the set of proper premises. We have evaluated our algorithms within an application related to refactoring of model variants. In this application an implicational base needs to be computed, and runtime is more crucial than minimal cardinality. In addition to the empirical tests, we provide heuristic evidence that an approach based on proper premises will also be beneficial for other applications. Finally, we show how our algorithms can be extended to an exploration algorithm that is based on proper premises.

Keywords formal concept analysis, proper premises

1 Introduction

For many years, computing the stem base has been the default method for extracting a small but complete set of implications from a formal context. There exist mainly two algorithms to achieve this [17,26], and both of them compute not only the implications from the stem base, but also concept intents. This is problematic as a context may have exponentially many

The author Daniel Borchmann has been supported by DFG Graduiertenkolleg 1763 (QuantLA)

U. Ryssel
Institute of Applied Computer Science, Technische Universität Dresden, Dresden, Germany, E-mail: uwe.ryssel@tu-dresden.de

F. Distel
Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany, E-mail: felix@tcs.inf.tu-dresden.de

D. Borchmann
Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany, E-mail: borch@tcs.inf.tu-dresden.de

concept intents. Recent theoretical results also suggest that improvements to this exponential worst-case complexity cannot be expected [10, 1].

In the early days of formal concept analysis an alternative to the stem base was developed, the base of *proper premises*. It has since been neglected as the stem base appeared superior since—unlike the base of proper premises—it has minimal cardinality. In this work we suggest to reconsider the base of proper premises. Just like pseudo-intents, which are used to obtain the stem base, proper premises yield a sound and complete set of implications. There are substantial arguments to reconsider using them. Existing methods for computing proper premises avoid computing concept intents, i.e. the main cause for the computational cost of stem base algorithms. Thus, in contexts with many concept intents they may have a clear advantage in runtime over the stem base algorithms.

Often, in applications, runtime is the limiting factor, not the size of the base. But even where minimal cardinality is a requirement, computing proper premises is worth considering, since there are methods to transform a base into the stem base in polynomial time [24, 28].

In this paper we present an algorithm for the fast computation of proper premises. It is based on three ideas. The first idea is to use a simple connection between proper premises and minimal hypergraph transversals. The problem of enumerating minimal hypergraph transversals is well-researched. Exploiting the link to proper premises allows us to use existing algorithms that are known to behave well in practice. A first, naïve algorithm iterates over all attributes and uses a black-box hypergraph algorithm to compute proper premises of each attribute.

A drawback when iterating over all attributes is that the same proper premise may be computed several times for different attributes. So we introduce a candidate filter in the second step: For each attribute m, the attribute set is filtered and proper premises are searched only among the candidate attributes. We show that this filtering method significantly reduces the number of multiple-computed proper premises while maintaining completeness. In a third step we exploit the fact that there are obvious redundancies within the proper premises. These can be removed by searching for proper premises only among the meet-irreducible attributes.

We argue that our algorithms are trivial to parallelize, leading to further speedups. Due to their incremental nature, parallelized versions of the stem base algorithms are not known to date.

We provide experimental results that support our claim that our algorithms perform well in practice. The application in which we test them is described in detail in Section 5.2. It deals with data-flow-oriented simulation models, such as MATLAB/Simulink, state diagrams, and diagrams of electrical networks. Generally, such models consist of blocks or elements and connections among them. Using techniques described in Section 5.2, a formal context can be obtained from such models. By computing an implication base of this context, dependencies among model artifacts can be uncovered. These can help to represent a large number of model variants in a structured way. In these contexts the number of concept intents is often close to the theoretical maximum. Here, attributes often occur together with their negated counterparts, and the concept lattice can contain several millions of elements.

Our empirical tests show highly significant improvements for the contexts obtained from the model refactoring application. For a sample context, where several hours where required to compute the stem base, runtime has dropped to fractions of a second. For contexts from other applications the improvements are not as impressive but still large.

In Section 5.1 we provide arguments that we can expect the number of concept intents to be larger than the number of proper premises in most contexts, assuming a uniform random distribution.

Another setting where proper premises may lead to a significant speedup is *attribute exploration* [18]. The aim of attribute exploration is to assist domain experts in completing knowledge that can be represented as formal contexts. During the exploration process, the domain expert is asked whether certain implications hold or not. If the domain expert refutes one of those implications, she is asked to provide a counterexample, which is then added to the formal context. The process terminates when no more implications exist that may be presented to the expert and the resulting set of implications completely describes the domain represented by the expert.

Attribute exploration is typically formulated using *pseudo-intents* and the Next-Closure [17, 18] algorithm. However, it is known [11,13] that in this setting it may take time exponential in the size of the currently known context to compute the next implication presented to the expert. Because of this, it may be worth considering alternative formulations of attribute exploration that do not use pseudo-intents. A first attempt to formulate attribute exploration using proper premises has been done in [27]. We shall give another variant of attribute exploration using proper premises in Section 6. It uses the same approaches as our enumeration algorithms for the stem base.

2 Preliminaries

We provide a short summary of the most common definitions in formal concept analysis (FCA). A *formal context* is a triple $\mathbb{K} = (G, M, I)$ where G is a set of objects, M a set of attributes, and $I \subseteq G \times M$ is a relation that expresses whether an object $g \in G$ has an attribute $m \in M$. If $A \subseteq G$ is a set of objects then A' denotes the set of all attributes that are shared among all objects in A, i.e., $A' = \{ m \in M \mid \forall g \in A\colon gIm \}$. Likewise, for some set $B \subseteq M$ we define $B' = \{ g \in G \mid \forall m \in B\colon gIm \}$. For objects $g \in G$, we may write g' instead of $\{ g \}'$, and likewise m' instead of $\{ m \}'$ for attributes $m \in M$. Pairs of the form (A, B) where $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$ are called *formal concepts*. Formal concepts of the form (m', m'') for some attribute $m \in M$ are called *attribute concepts* and are denoted by μm. We define the partial order $\leq$ on the set of all formal concepts of a context to be the subset order on the first component. The first component of a formal concept is called the *concept extent* while the second component is called the *concept intent*. The formal concepts of a formal context together with the order $\leq$ form a complete lattice, which is called the *concept lattice*. An example for a formal context together with its concept lattice can be seen in Figure 1. For conciseness the labeling in the lattice diagram is reduced. It can be understood in the following way: an object g has the attribute m iff the label m appears below the label g in the lattice diagram. For each $m \in M$ the attribute concept μm corresponds to the node labeled m in the lattice. For an object $g \in G$ and an attribute $m \in M$ we write $g \not\downarrow m$ if g' is maximal with respect to the subset order among all object intents which do not contain m.

Formal concept analysis provides methods to mine implicational knowledge from formal contexts. An *implication* is a pair (B_1, B_2) where $B_1, B_2 \subseteq M$, usually denoted by $B_1 \to B_2$. B_1 is then called the *premise of the implication* and B_2 is said to be the *conclusion of the implication*. We say that *the implication* $B_1 \to B_2$ *holds* in a context $\mathbb{K}$ if $B_1' \subseteq B_2'$. An implication $B_1 \to B_2$ *follows from a set of implications* $\mathscr{L}$ if for every context $\mathbb{K}$ in which all implications from $\mathscr{L}$ hold, $B_1 \to B_2$ also holds. This can alternatively be characterized in the

following way. Define

$$\begin{aligned}\mathscr{L}^1(A) &= A \cup \bigcup \{ Y \mid (X \to Y) \in \mathscr{L}, X \subseteq A \},\\ \mathscr{L}^i(A) &= \mathscr{L}^1(\mathscr{L}^{i-1}(A)) \qquad \text{for } i > 1,\\ \mathscr{L}(A) &= \bigcup_{i \in \mathbb{N}_{>0}} \mathscr{L}^i(A).\end{aligned}$$

Then an implication $X \to Y$ follows from the set $\mathscr{L}$ of implications if and only if $Y \subseteq \mathscr{L}(X)$. We write $\mathscr{L} \models (X \to Y)$ if and only if $X \to Y$ follows from $\mathscr{L}$.

We say that a set $\mathscr{L}$ of implications is *sound* for $\mathbb{K}$ if all implications from $\mathscr{L}$ hold in $\mathbb{K}$, and we say that $\mathscr{L}$ is *complete* for $\mathbb{K}$ if all implications that hold in $\mathbb{K}$ follow from $\mathscr{L}$. If $\mathscr{L}$ is sound and complete for $\mathbb{K}$, then $\mathscr{L}$ is said to be a *base*[1] for $\mathbb{K}$. It is called a *direct base*, if furthermore $\mathscr{L}^1(A) = \mathscr{L}(A)$ holds for all $A \subseteq M$.

There exists a sound and complete set of implications for each context which has minimal cardinality [19]. This is called the stem base. To define this base, we need to introduce the notion of *pseudo-intents*. These are sets $P \subseteq M$, such that $P \neq P''$ and for each pseudo-intent $Q \subsetneq P$, it is true that $Q'' \subseteq P$. Then the *stem base* of $\mathbb{K}$ is defined as

$$\{ P \to P'' \mid P \text{ pseudo-intent of } \mathbb{K} \}.$$

However, we can also explicitly describe another base of $\mathbb{K}$. Let $m \in M$ and $B \subseteq M$. Then B is called a *premise for* m if $m \in B'' \setminus B$. It is easy to see that the set

$$\mathscr{L} = \{ B \to B'' \mid B \subseteq M \text{ premise for some } m \in M \}$$

is also a sound and complete set of implications of $\mathbb{K}$, although it is quite large, as each subset $B \subseteq M$ is a premise for all $n \in B'' \setminus B$.

A smaller subsets of $\mathscr{L}$ that is still sound and complete can be obtained using *proper premises*. For a given set of attributes $B \subseteq M$, define $B^\bullet$ to be the set of those attributes in $M \setminus B$ that follow from B but not from a strict subset of B, i.e.,

$$B^\bullet = B'' \setminus \left(B \cup \bigcup_{S \subsetneq B} S'' \right).$$

B is called a *proper premise* if $B^\bullet$ is not empty. It is called a *proper premise for* $m \in M$ if $m \in B^\bullet$. One can show that B is a proper premise for m iff B is $\subseteq$-minimal among the premises of m. Furthermore, $\{ B \to B^\bullet \mid B \text{ proper premise} \}$ is sound and complete [18] and a direct base of $\mathbb{K}$. This set is called the *base of proper premises* of $\mathbb{K}$. Several alternative ways to define this base can be found in [4].

Example 1 We compute the proper premises in the context from Figure 1. Notice that a set $B \subseteq M$ and an attribute $m \in M$ satisfy $m \in B''$ iff $\bigwedge \{ \mu b \mid b \in B \} \leq \mu m$ holds in the concept lattice. Hence, $\{c,d,e\}$ is a premise for a. However, it is not $\subseteq$-minimal since $\{c,e\}$ and $\{d\}$ are also premises for a. The latter are $\subseteq$-minimal and therefore proper premises for a. The full base of proper premises of this context is

$$\begin{array}{lll} \{a,b\} \to \{d\}, & \{a,c\} \to \{b,d,e\}, & \{a,e\} \to \{b,c,d\},\\ \{b,c\} \to \{a,d,e\}, & \{b,e\} \to \{a,c,d\}, & \{c,d\} \to \{e\},\\ \{c,e\} \to \{a,b,d\}, & \{d\} \to \{a,b\}, & \{d,e\} \to \{c\}. \end{array}$$

[1] In the literature, sometimes what we call base is called *cover*, and what we call minimal base is simply called *base*. However, we shall not use this terminology in this work. In particular this means that our bases do not need to be minimal.

	a	b	c	d	e
1	×				
2		×			
3			×		
4	×	×		×	
5					×

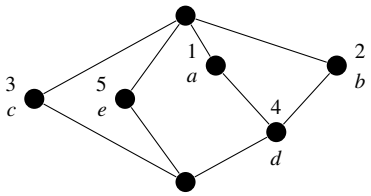

Fig. 1 A formal context and its concept lattice

3 Proper Premises as Minimal Hypergraph Transversals

We present a connection between proper premises and minimal hypergraph transversals, which forms the foundation for our enumeration algorithms. It has been exploited before in database theory to the purpose of mining functional dependencies from a database relation [25]. Implicitly, it has also been known for a long time within the FCA community. However, the term *hypergraph* has not been used in this context (cf. Prop. 23 from [18]). Beyond that, hypergraphs have previously been used for the related task of association rule mining [33]. An overview of how hypergraphs can be applied to data mining can be found in [20].

Let V be a finite set of vertices. A *hypergraph $\mathscr{H}$ on V* is simply a subset of the power set 2^V. Intuitively, each set $E \in \mathscr{H}$ represents an edge of the hypergraph, which, in contrast to classical graph theory, may be incident to more or less than two vertices. A set $S \subseteq V$ is called a *hypergraph transversal* of $\mathscr{H}$ if it intersects every edge $E \in \mathscr{H}$, i.e.,

$$\forall E \in \mathscr{H} \colon S \cap E \neq \emptyset.$$

$S \subseteq V$ is called a *minimal hypergraph transversal of $\mathscr{H}$* if it is minimal with respect to the subset order among all hypergraph transversals of $\mathscr{H}$. The *transversal hypergraph* of $\mathscr{H}$ is the set of all minimal hypergraph transversals of $\mathscr{H}$. It is denoted by $Tr(\mathscr{H})$. The problem of deciding for two hypergraphs $\mathscr{G}$ and $\mathscr{H}$ whether $\mathscr{H}$ is the transversal hypergraph of $\mathscr{G}$ is called TRANSHYP. The problem of enumerating all minimal hypergraph transversals of a hypergraph $\mathscr{G}$ is called TRANSENUM. Both problems are relevant to a large number of fields and therefore have been well-researched. TRANSHYP is known to be contained in CONP. Since it has been shown that TRANSHYP can be decided in quasi-polynomial time [16], it is not believed to be CONP-complete. Furthermore, it has been shown that it can be decided using only limited non-determinism [15]. For the enumeration problem it is not known to date whether an output-polynomial algorithm exists. However, efficient algorithms have been developed for several classes of hypergraphs [15, 7].

The following proposition can be found in [18] among others.

Proposition 1 *$P \subseteq M \setminus \{m\}$ is a premise of $m \in M$ iff*

$$(M \setminus g') \cap P \neq \emptyset$$

holds for all $g \in G$ with $g \not\mathrel{|} m$. P is a proper premise for m iff P is minimal (with respect to $\subseteq$) with this property.

We immediately obtain the following corollary.

Corollary 1 *P is a premise of m iff P is a hypergraph transversal of $\mathscr{H}^{\not|}_{\mathbb{K},m}$ where*

$$\mathscr{H}^{\not|}_{\mathbb{K},m} := \{(M \setminus \{m\}) \setminus g' \mid g \in G, g \not\mathrel{|} m\}.$$

The set of all proper premises of m is exactly the transversal hypergraph $Tr(\mathscr{H}^{\not|}_{\mathbb{K},m})$.

Algorithm 1 Naïve Algorithm for Enumerating All Proper Premises

Input: $\mathbb{K} = (G, M, I)$
$\mathscr{P} := \emptyset$
for all $m \in M$ **do**
  $\mathscr{P} := \mathscr{P} \cup Tr(\mathscr{H}^{\swarrow}_{\mathbb{K},m})$
end for
return $\mathscr{P}$

Table 1 Context from Figure 1 with $\swarrow$-Relation

	a	b	c	d	e
1	×	$\swarrow$		$\swarrow$	
2	$\swarrow$	×		$\swarrow$	
3	$\swarrow$	$\swarrow$	×	$\swarrow$	$\swarrow$
4	×	×	$\swarrow$	×	$\swarrow$
5	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$	×

Table 2 Context from Figure 1 inverted

	a	b	c	d	e
1		×	×	×	×
2	×		×	×	×
3	×	×		×	×
4			×		×
5	×	×	×	×	

Table 3 Context from Table 2 with Rows 1, 2, 3 and Column c Removed

	a	b	d	e
4				×
5	×	×	×	

A version of this corollary without the maximality condition (implied by $\swarrow$) on the object intents also exists.

Corollary 2 *The set of all proper premises of m is exactly the transversal hypergraph $Tr(\mathscr{H}^{\notin}_{\mathbb{K},m})$, where*

$$\mathscr{H}^{\notin}_{\mathbb{K},m} := \{(M \setminus \{m\}) \setminus g' \mid g \in G, m \notin g'\}.$$

In particular this proves that enumerating the proper premises of a given attribute m is polynomially equivalent to TRANSENUM. This can be exploited in a naïve algorithm for computing all proper premises of a formal context (Algorithm 1). Being aware of the link to hypergraph transversals, we can benefit from existing efficient algorithms for TRANSENUM in order to enumerate proper premises similar to what has been proposed in [25]. Of course, it is also possible to use other enumeration problems to which TRANSENUM can be reduced. Examples are the enumeration of prime implicants of Horn functions [4] and the enumeration of set covers.

Example 2 Assume we want to compute the proper premises for the attribute c in the context $\mathbb{K}$ from Figure 1. To see how the hypergraph $\mathscr{H}^{\swarrow}_{\mathbb{K},c}$ is obtained, consider first Table 1 which shows $\mathbb{K}$ with the $\swarrow$-relation added. Since only Rows 4 and 5 contain arrows in Column c only these two rows are relevant for the proper premises of c. Next, consider Table 3 which is obtained by first inverting the context (Table 2) and then removing the irrelevant rows as well as the Column c itself. Reading this context line by line yields the hypergraph $\mathscr{H}^{\swarrow}_{\mathbb{K},c} = \big\{\{e\}, \{a,b,d\}\big\}$. We obtain the proper premises of c as the minimal transversals of $\mathscr{H}^{\swarrow}_{\mathbb{K},c}$: they are $\{a,e\}$, $\{b,e\}$ and $\{d,e\}$.

4 Improvements to the Algorithm

4.1 Avoiding Duplicates using Candidate Sets

We can further optimize Algorithm 1 by reducing the search space. In the naïve algorithm proper premises are typically computed multiple times since they can be proper premises

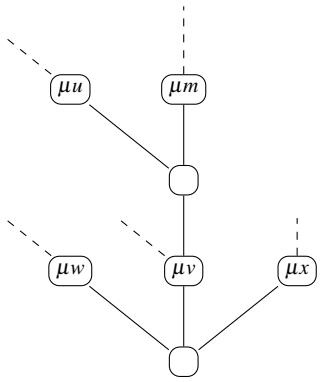

Fig. 2 On candidate sets

of more than one attribute. Our goal is to avoid this wherever possible. In Example 1 the set $\{c,e\}$ is a proper premise for the three attributes a, b and d. Therefore, Algorithm 1 will compute it three times, namely in the iterations for a, b and d.

The first idea is shown in Algorithm 2. There we introduce a *candidate set* C of relevant attributes, depending on the current attribute m. We claim now that we only have to search for minimal hypergraph transversals P of $\mathscr{H}^{\swarrow}_{\mathbb{K},m}$ with $P \subseteq C$. We provide some intuition for this idea.

Let us consider the snippet of a concept lattice as it is shown in Figure 2. We can see that in this case the set $\{w,x\}$ is a proper premise for both m and v. This implies that Algorithm 1 computes the set $\{w,x\}$ at least twice. More generally, we can observe that a proper premise P for an attribute m will be computed a second time in Algorithm 1 if

$$\bigwedge_{p\in P} \mu p \leq \mu v < \mu m$$

is true for some attribute v. The proper premise P is then recomputed in the iteration of v.

We now try to identify attributes that cause proper premises to be computed multiple times. Observe that in Figure 2 the attribute w satisfies the condition

$$\mu w \wedge \mu m \leq \mu v < \mu m. \tag{1}$$

Now, assume that P is a proper premise of m that contains w. Then we know that $m \in P''$, which is equivalent to

$$\bigwedge_{p\in P} \mu p \leq \mu m.$$

From $w \in P$ we obtain

$$\bigwedge_{p\in P} \mu p = \mu w \wedge \underbrace{\bigwedge_{p\in P} \mu p}_{\leq \mu m} \leq \mu w \wedge \mu m \leq \mu v.$$

Thus, P is also a proper premise for v. This shows that in the example any proper premise that contains w will be computed by Algorithm 1 at least twice: in the iteration for v and in the iteration for m. The redundancy is caused by w satisfying the condition (1). We therefore

Algorithm 2 A Better Algorithm for Enumerating All Proper Premises

Input: $\mathbb{K} = (G,M,I)$
$\mathscr{P} := \{\{m\} \mid m \in M, \{m\} \text{ is a proper premise of } \mathbb{K}\}$
for all $m \in M$ **do**
  $C := \{u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m \leq \mu v < \mu m\}$
  $\mathscr{P} := \mathscr{P} \cup \{P \subseteq C \mid P \text{ minimal hypergraph transversal of } \mathscr{H}^{\swarrow}_{\mathbb{K},m}\}$
end for
return $\mathscr{P}$

suggest to introduce in each iteration a candidate set of only those attributes that do not satisfy (1).

More formally, let us fix a formal context $\mathbb{K} = (G,M,I)$ and choose $m \in M$. In the iteration for m we search for proper premises only within the candidate set

$$C = \{u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m \leq \mu v < \mu m\}, \tag{2}$$

as shown in Algorithm 2. With this intuitive understanding in mind, we now turn to the proof of the correctness of Algorithm 2.

Lemma 1 *Algorithm 2 enumerates for a given formal context $\mathbb{K} = (G,M,I)$ all proper premises of $\mathbb{K}$.*

Proof Let P be a proper premise of $\mathbb{K}$ for the attribute m. P is a proper premise and therefore $m \in P''$ holds, which is equivalent to $\mu m \geq (P',P'')$. Let $c \in M$ be such that $\mu m \geq \mu c \geq (P',P'')$ and μc is minimal with this property. We claim that either $P = \{c\}$ or P is found in the iteration for c of Algorithm 2.

Suppose $c \in P$. Then $m \in \{c\}''$ follows from $\mu m \geq \mu c$. As a proper premise, P is minimal with the property $m \in P''$. It follows that $P = \{c\}$ and P is found by Algorithm 2 during the initialization.

Now suppose $c \notin P$. Consider

$$C := \{u \in M \setminus \{c\} \mid \nexists v \in M\colon \mu u \wedge \mu c \leq \mu v < \mu c\}.$$

We shall show that $P \subseteq C$. To see this, consider some $p \in P$. Then $p \neq c$ holds by assumption. Suppose that $p \notin C$, i.e., there is some $v \in M$ such that $\mu p \wedge \mu c \leq \mu v < \mu c$. Because of $p \in P$, $\mu p \geq (P',P'')$ and together with $\mu c \geq (P',P'')$ we have

$$(P',P'') \leq \mu p \wedge \mu c \leq \mu v < \mu c$$

in contradiction to the minimality of μc. This shows that $p \in C$ and all together $P \subseteq C$.

To complete the proof it remains to show that P is a minimal hypergraph transversal of $\{M \setminus \{g\}' \mid g \not\swarrow c\}$, i.e., that P is also a proper premise for c, not only for m. Consider $n \in P$. Assume $c \in (P \setminus \{n\})''$. Since $\{c\}$ implies m, then $P \setminus \{n\}$ would be a premise for m in contradiction to the minimality of P. Thus $c \notin (P \setminus \{n\})''$ holds for all $n \in P$ and therefore P is a proper premise for c. □

Example 3 As mentioned earlier the proper premise $\{c,e\}$ is a proper premise of three attributes, namely a, b and d. The naïve Algorithm 1 would therefore compute it three times.

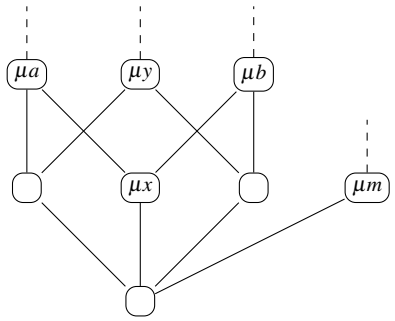

Fig. 3 On removing reducible attributes

Let us now compute the candidate set in the iteration of Algorithm 2 where $m = a$. It holds that

$$\begin{aligned} &\mu b \wedge \mu a \leq \mu d < \mu a, \\ &\mu c \wedge \mu a \leq \mu d < \mu a, \text{ and} \\ &\mu e \wedge \mu a \leq \mu d < \mu a. \end{aligned}$$

Thus b, c and e are not in the candidate set for a as defined in (2). In fact, the candidate set for a contains only one element: d. The proper premise $\{c,e\}$ will therefore not be computed in the iteration for a, nor will it be computed in the iteration for b. It will only appear in the iteration for d.

4.2 Irreducible Attributes

We go one step further and also remove attributes m from our candidate set C whose attribute concept μm is the meet of other attribute concepts $\mu x_1, \ldots, \mu x_n$, where $x_1, \ldots, x_n \in C$, i.e., $\mu m = \bigwedge_{i=1}^{n} \mu x_i$ and $\mu m \neq \mu x_i$ for $i = 1, \ldots, n$. Such an attribute m is called *(meet-)reducible*. This results in Algorithm 3 that no longer computes all proper premises, but a subset that still yields a complete implicational base. We show that we only have to search for proper premises P with $P \subseteq N$ where N is the set of irreducible attributes of $\mathbb{K}$.

To ease the presentation, let us assume for the rest of this paper that the formal context $\mathbb{K}$ is attribute-clarified, i.e. there are no two different attributes m,n in $\mathbb{K}$ such that $m' = n'$. Let us furthermore assume that $\emptyset'' = \emptyset$, i.e. there is no attribute that all objects have.

To obtain some intuition why reducible attributes may yield redundant proper premises let us consider the snippet of a concept lattice as it is given in Figure 3. We can see that the set $\{x,y\}$ is a proper premise for the attribute m. Notice that the attribute x is reducible since $\mu x = \mu a \wedge \mu b$. We can observe that the implication $\{x,y\} \to \{m\}$ already follows from the implications $\{x\} \to \{a,b\}$ and $\{a,b,y\} \to \{m\}$. The former is obtained from a singleton proper premise. In the latter the set $\{a,b,y\}$ is either a set of irreducible attributes or each reducible attribute can be replaced in the same way as for x. Hence, if we again handle singleton proper premise separately we are allowed to only consider the set of irreducible attributes when computing proper premises. This results in a significant speedup in the computation. The resulting algorithm is shown in Algorithm 3.

We are now going to describe this idea formally and prove its correctness.

Algorithm 3 Computing Enough Proper Premises

Input: $\mathbb{K} = (G, M, I)$
$\mathscr{P} := \{ \{m\} \mid m \in M, \{m\} \text{ is a proper premise of } \mathbb{K} \}$
$N := M \setminus \{ x \in M \mid \mu x = \bigwedge_{i=1}^{n} \mu x_i \text{ for an } n \in \mathbb{N} \text{ and } x_i \in M \setminus \{x\} \text{ for } 1 \leq i \leq n \}$
for all $m \in M$ **do**
  $C := \{ u \in N \setminus \{m\} \mid \nexists v \in M \colon \mu u \wedge \mu m \leq \mu v < \mu m \}$
  $\mathscr{P} := \mathscr{P} \cup \{ P \subseteq C \mid P \text{ minimal hypergraph transversal of } \mathscr{H}^{\swarrow}_{\mathbb{K},m} \}$
end for
return $\mathscr{P}$

Proposition 2 *Let m be an attribute and let P be a proper premise for m. Let $x \in P$, $n \in \mathbb{N}$, and for $1 \leq i \leq n$ let $x_i \in M$ be attributes satisfying*

- $m \notin \{x_1, \ldots, x_n\}$,
- $\mu x = \bigwedge_{i=1}^{n} \mu x_i$,
- $x_i \notin \emptyset''$ *for all* $1 \leq i \leq n$ *and*
- $\mu x < \mu x_i$ *for all* $1 \leq i \leq n$.

Then $\{x\}$ is a proper premise for all x_i and there exists a nonempty set $Y \subseteq \{x_1, \ldots, x_n\}$ such that $(P \setminus \{x\}) \cup Y$ is a proper premise for m.

Proof It is clear that $\{x\}$ is a proper premise for all x_i, since $x_i \in \{x\}''$ and $x_i \notin \emptyset''$. Define

$$Q_Y := (P \setminus \{x\}) \cup Y$$

for $Y \subseteq \{x_1, \ldots, x_n\}$. We choose $Y \subseteq \{x_1, \ldots, x_n\}$ such that Y is minimal with respect to $m \in Q_Y''$. Such a set exists, since $m \in ((P \setminus \{x\}) \cup \{x_1, \ldots, x_n\})''$ because of $\{x_1, \ldots, x_n\} \to \{x\}$. Furthermore, $Y \neq \emptyset$, since $m \notin (P \setminus \{x\})''$.

We now claim that Q_Y is a proper premise for m. Clearly $m \notin Q_Y$, since $m \notin Y$. For all $y \in Y$ it holds that $m \notin (Q_Y \setminus \{y\})''$ or otherwise minimality of Y would be violated. It therefore remains to show that $m \notin (Q_Y \setminus \{y\})''$ for all $y \in Q_Y \setminus Y = P \setminus \{x\}$.

$$\begin{aligned}(Q_Y \setminus \{y\})'' &= ((P \setminus \{x, y\}) \cup Y)'' \\ &\subseteq ((P \setminus \{y\}) \cup Y)'' \\ &= (P \setminus \{y\})''\end{aligned}$$

since $\{x\} \to Y$ and $x \in P \setminus \{y\}$. Since $m \notin (P \setminus \{y\})''$, we get $m \notin (Q_Y \setminus \{y\})''$ as required. In sum, Q_Y is a proper premise for m. □

Lemma 2 *Let N be the set of all meet-irreducible attributes of a context $\mathbb{K}$. Define*

$$\mathscr{P} = \{ X \subseteq M \mid |X| \leq 1, X \text{ proper premise} \} \cup \{ X \subseteq N \mid X \text{ proper premise} \}$$

Then the set $\mathscr{L} = \{ P \to P^{\bullet} \mid P \in \mathscr{P} \}$ is sound and complete for $\mathbb{K}$.

Proof Let m be an attribute and let P be a proper premise for m. If $P \notin \mathscr{P}$ then it follows that $P \not\subseteq N$. Thus we can find $y_1 \in P \setminus N$ and elements $x_1, \ldots, x_n \in M$ with $n \geq 1$ such that

- $m \notin \{x_1, \ldots, x_n\}$,
- $\mu y_1 = \bigwedge_{i=1}^{n} \mu x_i$,
- $x_i \notin \emptyset''$ for all $1 \leq i \leq n$ and
- $\mu x < \mu x_i$ for all $1 \leq i \leq n$.

By Proposition 2 we can find a proper premise P_1 such that $P \to \{m\}$ follows from $\{y_1\} \to \{x_1, \ldots, x_n\}$ and $P_1 \to \{m\}$. Clearly $\{y_1\} \in \mathscr{P}$, since all singleton proper premises are contained in $\mathscr{P}$. If $P_1 \notin \mathscr{P}$ then we can apply Proposition 2 again and obtain a new proper premise P_2, etc. To see that this process terminates consider the strict partial order $\prec$ defined as

$$P \prec Q \text{ iff } \forall q \in Q\colon \exists p \in P\colon \mu p < \mu q.$$

It is easy to see that with each application of Proposition 2 we obtain a new proper premise that is strictly larger than the previous with respect to $\prec$. Hence, the process must terminate. This yields a set $\mathscr{P}' = \{\{y_1\}, \ldots, \{y_k\}, P_k\} \subseteq \mathscr{P}$ such that $P \to \{m\}$ follows from $\{Q \to Q^\bullet \mid Q \in \mathscr{P}'\}$. Thus $\mathscr{L}$ is a sound and complete set of implications. □

Together with Lemma 1 this yields correctness of Algorithm 3.

Corollary 3 *The set of proper premises computed by Algorithm 3 yields a sound and complete set of implications for the given formal context.*

Example 4 In Figure 1 the attribute d is meet-reducible since $\mu d = \mu a \wedge \mu d$. In its first step Algorithm 3 will compute the set of all proper premises with at most one element:

$$\mathscr{P} = \big\{\{d\}\big\}.$$

In the second step, all meet-reducible attributes are removed and only the attributes in

$$N = \{a, b, c, e\}$$

are considered. After this Algorithm 3 proceeds exactly as Algorithm 2, but with N as the new set of attributes. It will find those proper premises that are subsets of N, i.e. $\{a,b\}$, $\{a,c\}$, $\{a,e\}$, $\{b,c\}$, $\{b,e\}$, and $\{c,e\}$. From the set of proper premises found in Example 1 only $\{c,d\}$ and $\{d,e\}$ are missing. The resulting base is still complete, since $\{c,d\} \to \{e\}$ follows from $\{d\} \to \{a,b\}$ and $\{a,c\} \to \{b,d,e\}$ (similarly for $\{d,e\} \to \{c\}$).

5 Evaluation

5.1 Computing Proper Premises Instead of Intents

In both the stem base algorithms and our algorithms, runtime can be exponential in the size of the input and output. In the classical case the reason is that the number of intents can be exponential in the size of the stem base [23]. In the case of our algorithms there are two reasons: the computation of proper premises is TRANSENUM-complete, and there can be exponentially many proper premises. The first issue is less relevant in practice because algorithms for TRANSENUM, while still exponential in the worst case, behave well for most instances.

To see that there can be exponentially many proper premises in the size of the stem base, let us look at the context $\mathbb{K}_n$ from Table 4 for some $n \geq 2$, consisting of two contranominal scales of dimension $n \times n$ and one attribute a with empty extent. It can be verified that the proper premises of the attribute a are exactly the sets of the form $\{m_i \mid i \in J\} \cup \{\bar{m}_i \mid i \notin J\}$ for some $J \subseteq \{1, \ldots, n\}$, while the only pseudo-intents are the singleton sets and $\{m_1, \ldots, m_n, \bar{m}_1, \ldots, \bar{m}_n\}$. Hence there are 2^n proper premises for a, while there are only $2n+2$ pseudo-intents.

Table 4 Context $\mathbb{K}_n$ with Exponentially Many Proper Premises

	m_1	…	m_n	$\bar{m}_1$	…	$\bar{m}_n$	a
g_1							
⋮		$\neq$			$\neq$		
g_n							

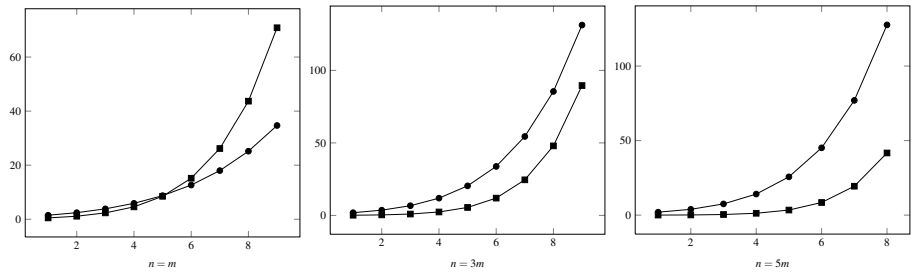

Fig. 4 Expected Number of Intents and Proper Premises for Certain Families of Formal Contexts

Next-Closure behaves poorly on contexts with many intents while our algorithms behave poorly on contexts with many proper premises. In order to provide evidence that our algorithm should behave better in practice we use formulae for the expectation of the number of intents and proper premises in a formal context that is chosen uniformly at random among all $n \times m$-contexts for fixed natural numbers n and m.[2] Derivations of these formulae can be found in [12].

The expected value for the number of intents in an $n \times m$-context is

$$\mathbb{E}_{\text{intent}} = \sum_{q=0}^{m} \binom{m}{q} \sum_{r=0}^{n} \binom{n}{r} 2^{-rq} (1-2^{-r})^{m-q} (1-2^{-q})^{n-r},$$

while the expected value for the number of proper premises for a fixed attribute a in an $n \times m$-context is

$$\mathbb{E}_{\text{pp}} = 2^{-n} \sum_{r=0}^{n} \binom{n}{r} \sum_{q=0}^{m-1} \binom{m}{q} q!\, 2^{-q^2} \sum_{\substack{(p_1,\dots,p_q)\in\mathbb{N}^q \\ 1\le p_1<\dots<p_q\le r}} \prod_{i=0}^{q} \left(1-2^{-q}(1+i)\right)^{p_{i+1}-p_i-1}.$$

Figure 4 shows the values of $m \cdot \mathbb{E}_{\text{pp}}$ (squares) and $\mathbb{E}_{\text{intent}}$ (bullets) for quadratic contexts and for contexts with $n = 3m$ and $n = 5m$. While there are more proper premises for quadratic contexts, less proper premises need to be computed for contexts with a large number of objects.

5.2 Applications in Model Refactoring

In Section 5.3 we shall see that the proper premise approach performs surprisingly well on contexts obtained from *model refactoring*. The goal of model refactoring is to restructure certain *models* that are used to describe and to design systems. One example of such models

[2] We ignore renaming of attributes and objects.

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	¬Feature 2	¬Feature 3	¬Feature 4	¬Feature 5	¬Feature 6
1	×					×	×		×	×	×	×	
2	×	×	×				×				×	×	×
3	×	×		×			×			×		×	×
4	×	×			×		×			×	×		×
5	×					×		×	×	×	×	×	
6	×	×	×					×			×	×	×
7	×	×		×				×		×		×	×
8	×	×			×			×		×	×		×

Feature	Artifacts
Feature 1	Block A, Block C
Feature 2	Block B, Line A to B, Line B to C
Feature 3	Block D, Line B to D
Feature 4	Block E, Line B to E
Feature 5	Block F, Line B to F
Feature 6	Line A to C
Feature 7	A.p = 2
Feature 8	A.p = 3

Fig. 5 A formal context for variant and features of data-flow models

are *data-flow-oriented* models. We briefly describe how formal contexts arise in this setting. Data-flow-oriented simulation models (e.g., MATLAB/Simulink models) are graph-like structures consisting of *artifacts* such as blocks, connections (lines) and parameter settings, which are used to model and run systems. Working with that kind of models will quickly result in many *variants* of similar models, which are difficult to manage.

One solution for that problem is to refactor these variants to one single configurable model. This model contains the union of all artifacts existing in any of the variants. An artifact that exists in more than one variant, which can be detected using matching algorithms, will be contained only once in the configurable model. Artifacts, which always co-occur in the variants, can be grouped to so-called features, so there is a 1:n-relation among features and artifacts. Complementary to the configurable model, a so-called feature model defines the valid combinations of this features in a form that is equivalent to a propositional formula with the features as variables.

To create a specific variant of the configurable model, the user has to select a number of features he wants in the resulting model. The validity of the given feature set is checked against the feature model, i.e., it is checked whether the set of selected features satisfies the corresponding propositional formula. Selected features will have the Boolean value *true* and not selected features will have the value *false*. If the feature set is valid, a generator copies those artifacts to the result model that are related to the selected features. Because of the restrictions in the feature model, the generator will always create valid models.

To create the needed feature models from a set of model variants automatically, as described in [29,30], the dependencies among the features or artifacts have to be identified in form of implications. Therefore, a formal context is built as shown in Figure 5: The model variants (1–8) form the object set and the features form the attribute set. Since the dependencies should also cover the relations to not selected features, a negated counterpart ¬Feature x is added as an additional attribute for some of the features, especially for blocks and lines. If a block or line exists in a variant, they are incident in the resulting context. For the negated counterparts the incidence is negated as well. The many-valued relation among variants and parameters is resolved by using parameter settings (i.e., pairs of parameters and their values) as attributes. For instance in Figure 5, the block A's parameter p creates one feature for each occurring value.

Typically, a context created from model variants contains many alternative features, i.e., features that never co-occur in the variants. In Figure 5, the features 3–6 and 7,8 are such groups of alternative features. Combined with the negated attributes, the number of concepts of the corresponding lattice will grow exponentially with the number of alternative features.

The context in Figure 5 contains only 46 concepts, but the data-flow model in Table 5 contains alternative feature groups with a size above 20 resulting in a lattice with millions of concepts. Also because of the negated attributes, the density of the context is high (usually greater than 0.3). A typical context size for this application is 20 objects and 60 attributes, from which a complete set of implications has to be calculated.

5.3 Experimental Comparison to Other Algorithms

We experimentally compare our proposed algorithm to other well-known ones. For this, we recall the algorithms we want to compare, briefly discuss some implementation details, and then present the achieved results.

Algorithms We compare the following implementations: *SB* which is an implementation of the stem base algorithm based on Next-Closure, *HT* which computes all proper premises as hypergraph transversals as in Algorithm 1, and *PP*, an implementation of Algorithm 3.

At first, comparing the algorithm SB, HT, PP may seem a bit suspicious, since they all compute different things from a given formal context. However, the purpose of our experiments is not to compare different ways to compute the same values. Instead, in our experiments we want to compare different ways to find bases of a given formal context. What we then want to compare are both the time needed to compute the respective base, and the size of the base. With respect to this goal, the algorithm SB, HT and PP can very well be compared, since these algorithms either directly compute a base (SB) or give the premises of a base (HT, PP).

Implementation An easy optimization we have made in HT and PP concerns parallelization. In all the listings we have presented so far, the iterations over the set of attributes in our formal context are independent of each other. It is natural to evaluate those iterations in parallel to improve the overall performance of our algorithms.

In our experiments we did not use special-purpose algorithms to compute the hypergraph transversals in HT and PP. Instead we have used an ad-hoc implementation that is based on backtracking and some simple heuristics [5]. Compared to advanced algorithms for TRANSENUM, this leaves room for further optimizations.

Data Sets Our test data sets can be divided into two categories, random and structured. For the tests on structured data we have chosen two data sets from the UCI Machine Learning Repository. The first data set is the testing data set of SPECT [9], which describes Single Proton Emission Computed Tomography (SPECT) images. This data set is given as a dyadic formal context with 187 objects, 23 attributes, and an approximate density of 0.38. The second one is a data set on Congressional Voting Records of the U.S. House of Representatives from 1984 [31]. It contains 435 objects, 16 attributes and is given as many valued context. It has been nominally scaled, resulting in a context with 50 attributes and an approximate density of 0.34.[3] The third structured data set originates from the application described in Section 5.2 and [30]. It has 26 objects, 79 attributes and an approximate density of 0.35.

The other part of testing data sets consists of randomly generated contexts. For this we fix the number n_G of objects, n_M of attributes and the density n_I of crosses. Then we generate

[3] Note that this is another scaling as the one in [26], so the times obtained there cannot be compared directly to ours.

Table 5 Behaviour on Structured Data

Context	SB		HT		PP	
	runtime	size	runtime	size	runtime	size
Data-Flow-Model	6.5 hrs	86	37 hrs	1 480	0.1 sec	86
SPECT	175 sec	1 778	16 sec	6 830	5.4 sec	6 830
Voting	13 hrs	18 572	6 hrs	140 032	17 min	140 032

Table 6 Behaviour on Random Data

Context	SB		HT		PP	
	runtime	size	runtime	size	runtime	size
$20 \times 40 \times 0.9$	75 sec	78	0.8 sec	988	1.4 sec	527
$20 \times 40 \times 0.8$	820 sec	879	4.3 sec	11 263	2.4 sec	9 223
$20 \times 40 \times 0.3$	8.9 sec	556	4.6 sec	3 780	2.5 sec	3 698
$20 \times 40 \times 0.2$	5.2 sec	386	2.2 sec	1 817	0.9 sec	1 478
$40 \times 20 \times 0.9$	17 sec	62	0.04 sec	105	0.04 sec	105
$40 \times 20 \times 0.8$	104 sec	920	1.5 sec	2 017	0.45 sec	2 017
$40 \times 20 \times 0.3$	1.6 sec	388	1 sec	1 258	0.7 sec	1 258
$40 \times 20 \times 0.2$	0.4 sec	173	0.4 sec	503	0.4 sec	503
$25 \times 25 \times 0.9$	17 sec	72	0.1 sec	154	0.04 sec	122
$25 \times 25 \times 0.8$	143 sec	565	1 sec	2 533	0.4 sec	2 533
$25 \times 25 \times 0.3$	1.2 sec	252	0.8 sec	1 231	0.6 sec	1 231
$25 \times 25 \times 0.2$	0.9 sec	226	0.34 sec	550	0.3 sec	533

for those three numbers one context of the given size $n_G \times n_M$ with an approximate density of n_I.

Experimental Results We have implemented all three algorithms as part of `conexp-clj`, a general-purpose FCA tool developed by one of the authors. The implementations itself are not highly optimized but rather prototypical, so the absolute running times of the test cases should not be taken as best possible. However, for comparing the three algorithms SB, HT, and PP, those implementations are sufficient and give a good impression on their performance. The experimental results (runtime and size of implication base) are given in Table 5 and Table 6.

As one can see from the results, HT most often runs faster then SB, but both are outperformed by PP. This can be seen most drastically with the Data-Flow-Model data sets, where PP only runs a fraction of a second whereas both SB and HT run for hours. The same occurs with the Voting data set. The same observation, although not that drastically, can also be seen with the randomly generated data sets.

The number of implications returned varies significantly not only between HT/PP and SB, but also between different runs of PP. Most often, HT and PP will return the same result, i.e., if the input context is attribute reduced. However, if it is not, the number of implications returned by PP may be significantly smaller than the overall number of proper premises, as one can see with the Data-Flow-Model data set, where the number of returned implications is the smallest possible.

However, most of the time the number of implications computed by HT and PP is much larger then the size of the stem base. The observed factors mostly range between 5 and 20. This might be a problem in practice, in particular if this factor is much higher. Therefore,

one has to consider a trade-off between the time one wants to spend on computing a sound and complete set of implications and on the size of this set of implications. The actual requirements of the particular application decide on the usefulness of the particular algorithm.

6 Attribute Exploration

Attribute exploration is an interactive formalism that can be used to obtain a sound and complete set of implications $\mathscr{L}$, even if the context $\mathbb{K}$ is initially incomplete. An attribute exploration system is assumed to have access to an incomplete context and an expert that has complete knowledge about the domain. In each iteration, an implication is computed that is then presented to the expert. The expert can either accept or refute it. If it is accepted, it is added to the set of implications $\mathscr{L}$. Otherwise, the expert is asked to provide a counterexample that is then added to the context.

Let us introduce some notation to facilitate the description of the algorithm. The formal context we use to start the exploration is called the *initial context* and the initial set of implications is called the *background knowledge*. During the exploration, at each step of the algorithm a formal context is known that originates from the initial context by adding all counterexamples obtained so far. This context is called the current *working context*. Likewise, the set of implications confirmed by the expert during the exploration process is said to be the set of *known implications*. Finally, when the exploration stops, the last working context is called the *final context*. We can imagine that during the exploration, although this context may not be known explicitly, the expert takes her counterexamples from an implicitly known context called the *background context* $\mathbb{K}_{\mathrm{BG}}$. We can then understand attribute exploration as the process of making the knowledge from the background context explicit.

Each time the current working context $\mathbb{K}$ is enlarged the set of implications that hold in $\mathbb{K}$ also changes. Algorithmically, the main challenge of attribute exploration is to avoid recomputing implications that have already been accepted by the expert. If the stem base is used, this can be done using the already mentioned Next-Closure algorithm. This algorithm enumerates implications (or rather their left-hand sides) in an order that extends the subset order. This order ensures that rejecting an implication has no influence on the implications that have been computed earlier.

When our algorithm for proper premises is used to compute the implications one has much less control over the order in which implications are obtained. Therefore, different arguments are needed. One attempt to this has already been done in [27]. In the following, we shall propose an alternative exploration algorithm using proper premises based on hypergraph transversals.

6.1 Incremental Computation of Proper Premises Using Berge Multiplication

Like in the traditional attribute exploration setting we assume that the expert does not make mistakes. Assume that she accepts an implication $P \to Q$ and, at a later point in the exploration, adds a counterexample resulting in a working context $\mathbb{K}_2$. Then we can assume that the expert does not contradict what she has stated earlier, i.e. $P \to Q$ still holds in $\mathbb{K}_2$. We show that in this case, P is still a proper premise of $\mathbb{K}_2$, meaning that once an implication is accepted its premise will remain a proper premise of the working context throughout the exploration process.

Lemma 3 *Let $\mathbb{K}_1 = (G_1, M, I_1)$ and $\mathbb{K}_2 = (G_2, M, I_2)$ be formal contexts such that $G_1 \subseteq G_2$ and $I_2 \cap (G_1 \times M) = I_1$. If P is a proper premise of $\mathbb{K}_1$ and $P \to P''_1$ holds in $\mathbb{K}_2$ then P is a proper premise of $\mathbb{K}_2$ and $P''_1 = P''_2$. Here $\cdot''_1$ (or $\cdot''_2$) denotes the derivations taken in $\mathbb{K}_1$ (or $\mathbb{K}_2$, respectively).*

Proof Notice that for any set of attributes B it holds that

$$B''_2 = \bigcap_{\substack{g \in G_2 \\ \forall m \in B:\ g I_2 m}} \{g\}'_2 \subseteq \bigcap_{\substack{g \in G_1 \\ \forall m \in B:\ g I_1 m}} \{g\}'_1 = B''_1. \tag{3}$$

In particular, this proves $P''_1 \supseteq P''_2$. Furthermore, since $P \to P''_1$ holds in $\mathbb{K}_2$ we get $P'_2 \subseteq (P''_1)'_2$, and therefore $P''_1 \subseteq P''_2$, yielding

$$P''_1 = P''_2. \tag{4}$$

Let $m \in M$ be such that P is a proper premise of m in $\mathbb{K}_1$. Then (4) proves that P is a premise of m in $\mathbb{K}_2$. It remains to prove minimality of P in $\mathbb{K}_2$. Let $A \subsetneq P$ be a strict subset of P. Since P is a proper premise of m in $\mathbb{K}_1$ we obtain that $m \notin A''_1$. Then $m \notin A''_2$ follows from (3). Hence no strict subset of P can be a premise of m in $\mathbb{K}_2$ and thus P is not only a premise, but a proper premise of m in $\mathbb{K}_2$. □

The above result is important, since it ensures that expert interaction is never redundant. In the following we will examine the algorithmic behaviour of proper premise based attribute exploration. We will use expressions like "The expert extends $\mathbb{K}_1$ by a valid counterexample." to say that the context is extended in such a way that Lemma 3 is applicable.

6.1.1 Berge Multiplication and its Offspring

So far, we have always treated the hypergraph transversal algorithms as a black box. However, for attribute exploration not all hypergraph transversal algorithms work equally well. For example the algorithm by Fredman and Khachiyan [16] has been shown to have a short total runtime in practice. However, it uses a divide and conquer approach which results in it returning all the solutions at once at the end of its runtime. This means that in attribute exploration one would have to wait for this algorithm to terminate until the first question can be presented to the expert.

In order for a hypergraph transversal algorithm to be a suitable algorithm for attribute exploration it should enumerate hypergraph transversals sequentially with a short delay. Such algorithms are preferable over those that return the complete set of algorithms at the end of the runtime, even if their total runtime were shorter. One of these algorithms is *Berge Multiplication* [3].

We define for two hypergraphs $\mathscr{G}$ and $\mathscr{H}$ their *edgewise union* $\mathscr{G} \vee \mathscr{H}$ as

$$\mathscr{G} \vee \mathscr{H} := \{ g \cup h \mid g \in \mathscr{G}, h \in \mathscr{H} \}.$$

Furthermore, for a set S of sets let us denote by $\min(S)$ all $\subseteq$-minimal sets in S, i. e.

$$\min(S) := \{X \in S \mid \nexists Y \in S : Y \subsetneq X\}.$$

Then the following statement holds, which can be found in [21], where it has been taken from [3].

Algorithm 4 Compute all minimal hypergraph transversals using Berge Multiplication

Input: $\mathscr{H}$
$\mathscr{T} := \{\emptyset\}$
for all $E \in \mathscr{H}$ **do**
  $\mathscr{T} := \min(\mathscr{T} \vee \{\{v\} \mid v \in E\})$
end for
return $\mathscr{T}$

Algorithm 5 Algorithm 1 with Berge Multiplication

Input: $\mathbb{K} = (G, M, I)$
$\mathscr{P} := \emptyset$
for all $m \in M$ **do**
  $\mathscr{E} := \mathscr{H}^{\notin}_{\mathbb{K},m}$
  $\mathscr{T} := \{\emptyset\}$
  for all $E \in \mathscr{E}$ **do**
    $\mathscr{T} := \min(\mathscr{T} \vee \{\{v\} \mid v \in E\})$
  end for
  $\mathscr{P} := \mathscr{P} \cup \mathscr{T}$
end for
return $\mathscr{P}$

Lemma 4 *Let $\mathscr{G}, \mathscr{H}$ be two finite hypergraphs. Then*

$$Tr(\mathscr{G} \cup \mathscr{H}) = \min(Tr(\mathscr{G}) \vee Tr(\mathscr{H})).$$

From this lemma it is now easy to obtain an algorithm that computes all hypergraph transversals of a finite hypergraph $\mathscr{H}$. The idea is, that for a hypergraph $\{E\}$ consisting of a single edge E the minimal hypergraph transversals are exactly the singleton sets $\{e\}$, $e \in E$. In Algorithm 4 edges are sequentially added by applying Lemma 4 until the complete transversal hypergraph of $\mathscr{H}$ is obtained. Algorithm 5 shows how Berge Multiplication can be used instead of the black box algorithm in Algorithm 1.

Despite its simplicity it has long been an open question whether Berge Multiplication can enumerate the minimal hypergraph transversals of $\mathscr{H}$ in output-polynomial time. In [32], Takata was able to show that indeed Berge Multiplication does not run in output-polynomial time. More precisely, Takata gave an example of a family of hypergraphs such that the minimal runtime of Berge Multiplication is at best $n^{\Omega(\log\log n)}$, where n is the size of the corresponding output.

On the other hand, in [8] Boros et. al. show that Berge Multiplication can be used to obtain all minimal hypergraph transversal of $\mathscr{H}$ in time $n^{\sqrt{n}}$, where again n denotes the size of the output. For this, the ordering of the edges of $\mathscr{H} = \{e_1, \ldots, e_n\}$ is significant but Boros et. al. were also able to show that the optimal permutation of the edges can be found in polynomial time.

There are a number of variations of the simple Berge Multiplication algorithm, as the DL-algorithm by Dong and Li [14], the BMR-algorithm by Bailey, Manoukian and Ramamohanarao [2] and the KS-algorithm by Kavvadias and Stavropoulos [22]. All these algorithms apply certain heuristics to speed up the multiplication step. However, all these algorithms have worst case runtime at least $n^{\Omega(\log\log n)}$ with n the size of the output [21].

6.1.2 A Naïve Exploration Algorithm using Proper Premises

We can now formulate a first version of our attribute exploration algorithm using proper premises. Let $\mathbb{K}$ be the initial context, $\mathscr{L}$ be the background knowledge and $\mathscr{E} = \mathscr{H}^{\notin}_{\mathbb{K},m}$.

Algorithm 6 Attribute Exploration using Proper Premises and Berge Multiplication

Input: $\mathbb{K} = (G, M, I)$
$\mathscr{L} := \emptyset$
for all $m \in M$ **do**
  $\mathscr{E} := \mathscr{H}^{\notin}_{\mathbb{K},m}$
  $\mathscr{T} := \{\emptyset\}$
  while there exists $E \in \mathscr{E}$ **do**
    $\mathscr{T} := \min(\mathscr{T} \vee \{\{v\} \mid v \in E\})$
    $\mathscr{E} := \mathscr{E} \setminus \{E\}$
    while there exists $Q \in \mathscr{T}$ with $\mathscr{L} \not\models (Q \to Q'')$ **do**
      if expert confirms $Q \to Q''$ **then**
        $\mathscr{L} := \mathscr{L} \cup \{Q \to Q''\}$
      else
        ask expert for valid counterexample g
        if $m \notin g'$ **then**
          $\mathscr{E} := \mathscr{E} \cup \{M \setminus g'\}$
        end if
      end if
    end while
  end while
end for
return $\mathscr{L}$

Intuitively, the algorithm iterates through all attributes m and computes proper premises P of m by successively considering edges in $\mathscr{E}$ using Berge Multiplication. If then $\mathscr{L} \not\models P \to P''$ the implication $P \to P''$ is asked to the expert. If the implication is accepted it is added to the set $\mathscr{L}$ of known implications. If the implication is rejected then let g be a counterexample for $P \to \{m\}$. Then if $m \notin g'$, then the set $M \setminus g'$ is an edge in $\mathscr{H}^{\notin}_{\mathbb{K},m}$ and hence is added to $\mathscr{E}$. The process continues until there are no more edges in $\mathscr{E}$ are left, whereupon all proper premises for m of the background context are known. The exploration then continues by considering the remaining attributes in M until no more are left.

This process is formally presented in Algorithm 6. Note that in this algorithm, we describe expert interaction only informally as "expert confirms $Q \to Q''$" or "ask expert for valid counterexample," as it is usual in the literature. But it is also possible to describe this interaction more formally, as it has been done in [6]. We shall not do this here, however, as it is not necessary for our further considerations.

Lemma 5 *Upon termination of Algorithm 6, $\mathscr{L}$ is a base of the background context $\mathbb{K}_{\mathrm{BG}}$ and contains only implications of the form $P \to P''$ where P is a proper premise of $\mathbb{K}_{\mathrm{BG}}$.*

Proof Let $\mathbb{K}$ be the final context obtained upon termination of Algorithm 6. Outside the inner while-loop Algorithm 6 behaves exactly like Algorithm 5. Thus, it follows from correctness of Algorithm 5 that upon termination it will have enumerated all proper premises of $\mathbb{K}$ (more precisely all implications $P \to P''$ where P is a proper premise of $\mathbb{K}$).

It only remains to show that the proper premises of $\mathbb{K}$ are exactly the proper premises of $\mathbb{K}_{\mathrm{BG}}$. First, it follows directly from Lemma 3 that every proper premise P of $\mathbb{K}$ is also a proper premise of $\mathbb{K}_{\mathrm{BG}}$.

Now assume that P is a proper premise in $\mathbb{K}_{\mathrm{BG}}$ for some attribute m. By Corollary 2 this is equivalent to P being a minimal hypergraph transversal of $\mathscr{H}^{\notin}_{\mathbb{K}_{\mathrm{BG}},m}$. Since all counterexamples have been taken from $\mathbb{K}_{\mathrm{BG}}$ it is true that $\mathscr{H}^{\notin}_{\mathbb{K},m} \subseteq \mathscr{H}^{\notin}_{\mathbb{K}_{\mathrm{BG}},m}$ and thus P is also a hypergraph transversal for $\mathscr{H}^{\notin}_{\mathbb{K},m}$. It remains to show minimality of P among the hypergraph transversals of $\mathscr{H}^{\notin}_{\mathbb{K},m}$. Let $Q \subseteq P$ be a minimal hypergraph transversal of $\mathscr{H}^{\notin}_{\mathbb{K},m}$. Then Q is a proper

Algorithm 7 Querying the Hierarchy of Attribute Concepts

Input: $\mathbb{K} = (G, M, I)$
$\mathbb{K}_{\text{init}} := \mathbb{K}$
$\mathscr{L}_{\text{init}} := \emptyset$
{find singleton proper premises:}
for all $m \in M$ **do**
  while expert refutes $\{m\} \to \{m\}''$ **do**
    expert adds counterexample to $\mathbb{K}_{\text{init}}$
  end while
  if $\{m\} \neq \{m\}''$ **then**
    $\mathscr{L}_{\text{init}} := \mathscr{L}_{\text{init}} \cup \{\{m\} \to \{m\}''\}$
  end if
end for
{find irreducibles:}
for all $m \in M$ **do**
  while expert refutes $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$ **do**
    expert adds counterexample to $\mathbb{K}_{\text{init}}$
  end while
end for
return $\mathbb{K}_{\text{init}}, \mathscr{L}_{\text{init}}$

premise of $\mathbb{K}$ and thus $\mathscr{L} \models (Q \to Q'')$. By Lemma 3, Q is also a proper premise of $\mathbb{K}_{\text{BG}}$ and therefore a minimal hypergraph transversal of $\mathscr{H}^{\notin}_{\mathbb{K}_{\text{BG}},m}$. We know that P is minimal among the transversals of $\mathscr{H}^{\notin}_{\mathbb{K}_{\text{BG}},m}$, which implies $P = Q$. This proves that P is also minimal among the transversals of $\mathscr{H}^{\notin}_{\mathbb{K},m}$, i.e. P is a proper premise of the working context $\mathbb{K}$.

We have thus shown that Algorithm 6 enumerates the base of proper premises of the final working context $\mathbb{K}$ and that this is the same as the base of proper premises of the background context $\mathbb{K}_{\text{BG}}$.

6.2 Using the improvements

While Algorithm 6 shows a way to perform attribute exploration using proper premises, it does not make use of the two improvements. In order to benefit from the performance gain achieved by these two ideas we need to implement them in our exploration algorithms.

6.2.1 Querying the hierarchy of attribute concepts

Both Algorithm 2 and Algorithm 3 require that all singleton proper premises be computed in a first step. In attribute exploration the context $\mathbb{K}$ is initially incomplete. It is therefore not sufficient to simply compute all singleton proper premises of $\mathbb{K}$, because we do not know whether they will remain proper premises when $\mathbb{K}$ is extended. However, if $\{m\}$ is a proper premise of $\mathbb{K}$ and the expert confirms the implication $\{m\} \to \{m\}''$ then Lemma 3 ensures that $\{m\}$ remains a premise when $\mathbb{K}$ is extended.

In Algorithm 3 one needs to know which attribute concepts are meet-irreducible. Unfortunately, it is possible that an attribute concept μm that is meet-reducible in the working context $\mathbb{K}$ is no longer meet-reducible in an extension of $\mathbb{K}$. We can find out for certain if the reducible attribute m can be removed by presenting the implication $\{n \in \{m\}'' \mid \mu m < \mu n\} \to \{m\}$ to the expert. We shall see that m remains meet-reducible in all extensions of $\mathbb{K}$ if this implication is accepted. Algorithm 7 shows how these two initial queries are performed.

Lemma 6 *Let $\mathbb{K}_{\text{init}} = (G_{\text{init}}, M, I_{\text{init}})$ and $\mathscr{L}_{\text{init}}$ be the context and implication set obtained by Algorithm 7. Let $\mathbb{K} = (G, M, I)$ be a formal context such that $G_{\text{init}} \subseteq G$ and $I \cap (G_{\text{init}} \times M) = I_{\text{init}}$. Let $m \in M$ be an attribute.*

If all implications from $\mathscr{L}_{\text{init}}$ hold in $\mathbb{K}$ then

- *$\{m\}$ is a proper premise of $\mathbb{K}$, and*
- *$\mu m \leq \mu n$ iff $\mu m \leq_{\text{init}} \mu n$, where $\leq$ and $\leq_{\text{init}}$ denote the order in the concept lattice of $\mathbb{K}$ and $\mathbb{K}_{\text{init}}$, respectively, and*
- *μm is meet-reducible in $\mathbb{K}$ iff μm is meet-reducible in $\mathbb{K}_{\text{init}}$.*

Proof The claim that $\{m\}$ is a proper premise of $\mathbb{K}$ follows immediately from Lemma 3 and the fact that for all proper premises $\{m\}$ the implication $\{m\} \to \{m\}''$ is contained in $\mathscr{L}_{\text{init}}$.

In particular, we also obtain from Lemma 3 that $\{m\}'' = \{m\}''^{\text{init}}$ holds for all $m \in M$, i.e. after the first for-loop has terminated, the intents of attribute concepts remain fixed. It is an easy result from basic FCA that $\mu m \leq \mu n$ iff $n \in \{m\}''$. Hence, together with the intents the hierarchy of attribute concepts also remains fixed:

$$\mu m \leq \mu n \iff \mu m \leq_{\text{init}} \mu n.$$

This proves the second claim.

We show that μm is meet-reducible in $\mathbb{K}$ iff $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$ holds in $\mathbb{K}$. On the one hand it holds that

$$\begin{aligned}\{n \in m'' \mid \mu m < \mu n\} \to \{m\} \text{ holds in } \mathbb{K} &\iff \{n \in m'' \mid \mu m < \mu n\}' \subseteq \{m\}' \\ &\iff \bigwedge\{\mu n \mid n \in m'', \mu m < \mu n\} \leq \mu m \iff \bigwedge\{\mu n \mid n \in m'', \mu m < \mu n\} = \mu m.\end{aligned}$$

This shows that μm is meet-reducible if $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$ holds in $\mathbb{K}$. On the other hand, if μm is meet-reducible then there is a set $S \subseteq \{n \in m'' \mid \mu m < \mu n\}$ such that $\bigwedge\{\mu n \mid n \in S\} = \mu m$. By the same arguments as above, this is equivalent to $S' \subseteq \{m\}'$ and together with $\{n \in m'' \mid \mu m < \mu n\}' \subseteq S'$ implies that $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$ holds in $\mathbb{K}$.

By asking the expert to confirm or refute each of the implications of the form $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$, we can be sure that $\{n \in m'' \mid \mu m < \mu n\} \to \{m\}$ holds in $\mathbb{K}$ iff it holds in $\mathbb{K}_{\text{init}}$. The above equivalence then shows that μm is irreducible in $\mathbb{K}$ iff it is irreducible in $\mathbb{K}_{\text{init}}$. □

6.2.2 Candidate Sets

Adapting the restriction to candidate sets to an exploration setting is not straightforward. The problem is that an attribute u that is not a candidate for an attribute m in $\mathbb{K}_{\text{init}}$ can still be a candidate in $\mathbb{K}_{\text{BG}}$, as is illustrated by the following example.

Consider the background context $\mathbb{K}_{\text{BG}}$ from Table 7. Assume that after the preprocessing step of Algorithm 7 we obtain a context $\mathbb{K}_{\text{init}}$ as shown in Table 8 and the set of implications $\mathscr{L}_{\text{init}} = \{\{v\} \to \{m\}\}$. Suppose we want to compute the proper premises for m. We compute the candidate set

$$C_{\text{init}} = \{u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m \leq_{\text{init}} \mu v <_{\text{init}} \mu m\},$$

which is empty, as can easily be seen from the concept lattice in Figure 7. As a consequence, we fail to compute the proper premise $\{u, w\}$. However, in the concept lattice of $\mathbb{K}_{\text{BG}}$ both u and w are candidates for m, and $\{u, w\}$ is indeed a proper premise for m (cf. Figure 6).

Table 7 Context $\mathbb{K}_{\mathrm{BG}}$

	m	u	v	w
A	×			
B		×		
C	×		×	
D				×
E	×	×		×

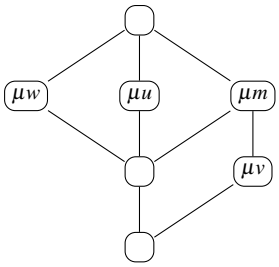

Fig. 6 Concept Lattice of $\mathbb{K}_{\mathrm{BG}}$

Table 8 Context $\mathbb{K}_{\mathrm{init}}$

	m	u	v	w
A	×			
B		×		
C	×		×	
D				×

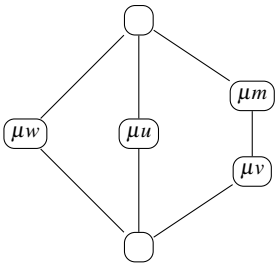

Fig. 7 Concept Lattice of $\mathbb{K}_{\mathrm{init}}$

Notice, that the above problem does not arise if we compute the proper premises for v before we compute those for m. Then the expert would first be asked whether $\{m,u\} \to \{v\}$ holds, forcing her to add E as the counterexample.

The problems from the above example can be easily avoided by fixing the order in which the attributes are processed. We can select any order $<$ on M that satisfies the condition

$$\mu m <_{\mathrm{init}} \mu n \qquad \text{implies} \qquad m < n.$$

An exploration algorithm that uses candidate sets and this order on the attributes is outlined in Algorithm 8.

If m is the first attribute with respect to this order, then μm is minimal among all attribute concepts, and therefore the candidate set for m is the full set $M \setminus \{m\}$. This means that all proper premises for m are known after the first iteration. Remember, that in Algorithm 2 each proper premise of m is obtained either in the iteration of m or in the iteration of some v with $\mu v < \mu m$. Thus, induction shows that for all attributes n the following holds: After the attribute n has been processed

- $\mathscr{L}$ contains the implication $P \to P''$ for every proper premise P of n, and
- for every implication of the form $S \to \{n\}$ that does not hold in $\mathbb{K}_{\mathrm{BG}}$ a counterexample has been added to $\mathbb{K}$.

This property ensures that in each iteration the candidates that would be computed in $\mathbb{K}_{\mathrm{BG}}$ are exactly the candidates computed for $\mathbb{K}$, as can be seen from the following lemma.

Lemma 7 *Assume that we have allowed Algorithm 8 to run until the beginning of the iteration for an attribute m. Let $\mathbb{K}_m$ be the current context obtained in this iteration and $\mathbb{K}_{\mathrm{BG}}$ the background context. Let $\leq_m$ and $\leq_{\mathrm{BG}}$ denote the orders in their respective concept lattices. Then*

$$\begin{aligned} C_{\mathrm{BG}} &= \{u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m \leq_{\mathrm{BG}} \mu v <_{\mathrm{BG}} \mu m\} \\ &= \{u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m \leq_m \mu v <_m \mu m\} = C_m \end{aligned} \quad (5)$$

Algorithm 8 Attribute Exploration Algorithm using Candidate Sets

Input: $\mathbb{K} = (G, M, I)$
Use Algorithm 7 to compute $\mathbb{K}_{\text{init}}$ and $\mathscr{L}_{\text{init}}$
$\mathscr{L} := \mathscr{L}_{\text{init}}$, $\mathbb{K} = \mathbb{K}_{\text{init}}$
fix an order $m_1 < \cdots < m_n$ on M such that $\mu m_i <_{\text{init}} \mu m_j$ implies $m_i < m_j$
for $m := m_1$ to m_n **do**
  $C := \{ u \in M \setminus \{m\} \mid \nexists v \in M\colon \mu u \wedge \mu m <_{\text{init}} \mu v <_{\text{init}} \mu m \}$
  $\mathscr{E} := \{ E \cap C \mid E \in \mathscr{H}^{\not\in}_{\mathbb{K},m} \}$
  $\mathscr{T} := \{\emptyset\}$
  while there exists $E \in \mathscr{E}$ **do**
    $\mathscr{T} := \min(\mathscr{T} \vee \{\{v\} \mid v \in E\})$
    $\mathscr{E} := \mathscr{E} \setminus \{E\}$
    while there exists $Q \in \mathscr{T}$ with $\mathscr{L} \not\models (Q \to Q'')$ **do**
      if expert confirms $Q \to Q''$ **then**
        $\mathscr{L} := \mathscr{L} \cup \{ Q \to Q'' \}$
      else
        ask expert for valid counterexample g
        if $m \notin g'$ **then**
          $\mathscr{E} := \mathscr{E} \cup \{C \setminus g'\}$
        end if
      end if
    end while
  end while
end for
return $\mathscr{L}$

Proof From Lemma 6 we obtain that

$$\mu v <_{\text{BG}} \mu m \iff \mu v <_{\text{init}} \mu m \iff \mu v <_m \mu m. \tag{6}$$

Furthermore, it is a basic fact from FCA that $\mu u \wedge \mu m \leq_{\text{BG}} \mu v$ iff $\{u,m\}'^{\text{BG}} \subseteq \{v\}'^{\text{BG}}$, i.e. iff $\{u,m\} \to \{v\}$ holds in $\mathbb{K}_{\text{BG}}$. If $\mu v <_m \mu m$ then we know that v has been processed in an earlier iteration, and therefore all implications of the form $S \to \{v\}$ hold in $\mathbb{K}_m$ iff they hold in $\mathbb{K}_{\text{BG}}$. Thus we obtain for all $v \in M$ with $\mu v <_m \mu m$

$$\begin{aligned} \mu u \wedge \mu m \leq_{\text{BG}} \mu v &\iff \{u,m\} \to \{v\} \text{ holds in } \mathbb{K}_{\text{BG}} \\ &\iff \{u,m\} \to \{v\} \text{ holds in } \mathbb{K}_m \iff \mu u \wedge \mu m \leq_m \mu v. \end{aligned} \tag{7}$$

From (6) and (7) we obtain the following equivalences.

$$\begin{aligned} u \notin C_m &\iff \exists v \in M\colon \mu u \wedge \mu m \leq_m \mu v <_m \mu m \\ &\iff \exists v \in M\colon \mu u \wedge \mu m \leq_{\text{BG}} \mu v <_{\text{BG}} \mu m \\ &\iff u \notin C_{\text{BG}}. \end{aligned}$$

This proves $C_m = C_{\text{BG}}$. □

Lemma 7 shows that if we run Algorithm 2 on the background context we obtain the same candidate sets as if we run Algorithm 8 with a subcontext as input. Correctness of Algorithm 8 then follows immediately from correctness of Algorithm 2 (Lemma 1).

Corollary 4 *Upon termination of Algorithm 8 with a subcontext $\mathbb{K}$ of $\mathbb{K}_{\text{BG}}$ as input the set $\mathscr{L}$ will contain all implications of the form $P \to P''$ where P is a proper premise in $\mathbb{K}_{\text{BG}}$.*

Algorithm 9 Attribute Exploration Algorithm without Reducible Attributes

Input: $\mathbb{K} = (G, M, I)$
Use Algorithm 7 to compute $\mathbb{K}_{\text{init}}$ and $\mathscr{L}_{\text{init}}$
$\mathscr{L} := \mathscr{L}_{\text{init}}$, $\mathbb{K} := \mathbb{K}_{\text{init}}$
$N := M \setminus \{ x \in M \mid \mu x = \bigwedge_{i=1}^{n} \mu x_i \text{ for an } n \in \mathbb{N} \text{ and } x_i \in M \setminus \{x\} \text{ for } 1 \leq i \leq n \}$
fix an order $m_1 < \cdots < m_n$ on M such that $\mu m_i <_{\text{init}} \mu m_j$ implies $m_i < m_j$
for $m := m_1$ to m_n **do**
  $C := \{ u \in N \setminus \{m\} \mid \nexists v \in M \colon \mu u \wedge \mu m <_{\text{init}} \mu v <_{\text{init}} \mu m \}$
  $\mathscr{E} := \{ E \cap C \mid E \in \mathscr{H}^{\not\in}_{\mathbb{K},m} \}$
  $\mathscr{T} := \{\emptyset\}$
  while there exists $E \in \mathscr{E}$ **do**
    $\mathscr{T} := \min(\mathscr{T} \vee \{\{v\} \mid v \in E\})$
    $\mathscr{E} := \mathscr{E} \setminus \{E\}$
    while there exists $Q \in \mathscr{T}$ with $\mathscr{L} \not\models (Q \to Q'')$ **do**
      if expert confirms $Q \to Q''$ **then**
        $\mathscr{L} := \mathscr{L} \cup \{ Q \to Q'' \}$
      else
        ask expert for valid counterexample g
        if $m \notin g'$ **then**
          $\mathscr{E} := \mathscr{E} \cup \{C \setminus g'\}$
        end if
      end if
    end while
  end while
end for
return $\mathscr{L}$

6.2.3 Removing Reducibles

Because of Lemma 6 we can be sure that once $\mathbb{K}_{\text{init}}$ has been computed the set of irreducible attributes does not change anymore during the exploration, and is in fact the same as the set of irreducible attributes in $\mathbb{K}_{\text{BG}}$. Hence, we can directly extend Algorithm 8 by restricting the search space to meet-irreducible attributes. This is shown in Algorithm 9. Correctness of Algorithm 9 is an immediate consequence of correctness of Algorithm 3 and Algorithm 9.

Corollary 5 *Upon termination of Algorithm 9 with a subcontext $\mathbb{K}$ of $\mathbb{K}_{\text{BG}}$ as input the set $\mathscr{L}$ will contain all implications of the form $P \to P''$ where P is a proper premise in $\mathbb{K}_{\text{BG}}$.*

7 Conclusion

In this paper we have shown that, while the stem base has minimal cardinality, the base of proper premises can often be computed more efficiently. Very simple optimizations in the hypergraph based algorithms yield large gains in performance. Therefore it appears promising that using more sophisticated approaches even larger performance gains can be achieved.

We have evaluated it both on random, artificially generated contexts as well as on contexts from a practical application. Our evaluation has shown our algorithms to be faster than existing algorithms on these particular datasets. This suggests that our approach may also be better for other datasets, especially if these datasets contain a large number of intents.

It should be mentioned that in a setting where minimal cardinality is crucial an additional minimization step is required. In this case the performance gain may be smaller. However, in applications like model refactoring minimal cardinality is only of secondary interest.

We have also shown how a proper premise based exploration can be performed. Previously, only high-level descriptions of proper premise based attribute exploration existed. We have provided a detailed algorithmic description and we have shown how the improvements from the first part of the paper can be applied in an attribute exploration setting.

References

1. Babin, M., Kuznetsov, S.: Recognizing pseudo-intents is coNP-complete. In: M. Kryszkiewicz, S. Obiedkov (eds.) Proc. of the 7th Int. Conf. on Concept Lattices and Their Applications, vol. 672. CEUR Workshop Proceedings (2010)
2. Bailey, J., Manoukian, T., Ramamohanarao, K.: A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In: ICDM, pp. 485–488. IEEE Computer Society (2003)
3. Berge, C.: Hypergraphs – combinatorics of finite sets. North-Holland, Amsterdam (1989)
4. Bertet, K., Monjardet, B.: The multiple facets of the canonical direct unit implicational basis. Theoretical Computer Science **411**(22–24), 2155–2166 (2010)
5. Borchmann, D.: conexp-clj — a general-purpose tool for formal concept analysis. See http://www.math.tu-dresden.de/ borch/conexp-clj/
6. Borchmann, D.: A General Form of Attribute Exploration. LTCS-Report 13-02, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2013). See http://lat.inf.tu-dresden.de/research/reports.html.
7. Boros, E., Elbassioni, K., Gurvich, V., Khachiyan, L.: An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension. Parallel Processing Letters **10**, 253–266 (2000)
8. Boros, E., Elbassioni, K., Makino, K.: On berge multiplication for monotone boolean dualization. In: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I, ICALP '08, pp. 48–59. Springer-Verlag, Berlin, Heidelberg (2008)
9. Cios, K.J., Kurgan, L.A., Goodenday, L.S.: UCI Machine Learning Repository: SPECT Heart Data Set (2001). URL `http://archive.ics.uci.edu/ml/datasets/SPECT+Heart`
10. Distel, F.: Hardness of enumerating pseudo-intents in the lectic order. In: B. Sertkaya, L. Kwuida (eds.) Proc. of the 8th Int. Conf. on Formal Concept Analysis, *Lecture Notes in Artificial Intelligence*, vol. 5986, pp. 124–137. Springer (2010)
11. Distel, F.: Hardness of enumerating pseudo-intents in the lectic order. In: L. Kwuida, B. Sertkaya (eds.) ICFCA, *Lecture Notes in Computer Science*, vol. 5986, pp. 124–137. Springer (2010)
12. Distel, F., Borchmann, D.: Expected numbers of proper premises and concept intents. Preprint, Institut für Algebra, TU Dresden (2011)
13. Distel, F., Sertkaya, B.: On the complexity of enumerating pseudo-intents. Discrete Applied Mathematics **159**(6), 450–466 (2011)
14. Dong, G., Li, J.: Mining border descriptions of emerging patterns from dataset pairs. Knowl. Inf. Syst. **8**(2), 178–202 (2005)
15. Eiter, T., Gottlob, G., Makino, K.: New results on monotone dualization and generating hypergraph transversals. SIAM J. Comput. **32**, 514–537 (2003)
16. Fredman, M.L., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. Journal of Algorithms **21**(3), 618 – 628 (1996)
17. Ganter, B.: Two basic algorithms in concept analysis. Preprint 831, Fachbereich Mathematik, TU Darmstadt, Darmstadt, Germany (1984)
18. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1999)
19. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. Sci. Humaines **95**, 5–18 (1986)
20. Gunopulos, D., Mannila, H., Khardon, R., Toivonen, H.: Data mining, hypergraph transversals, and machine learning (extended abstract). In: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, PODS '97, pp. 209–216. ACM, New York, NY, USA (1997). DOI 10.1145/263661.263684. URL `http://doi.acm.org/10.1145/263661.263684`
21. Hagen, M.: Algorithmic and Computational Complexity Issues of MONET. Ph.D. thesis, Friedrich-Schiller-Universität Jena (2008)
22. Kavvadias, D.J., Stavropoulos, E.C.: An efficient algorithm for the transversal hypergraph generation. J. Graph Algorithms Appl. **9**(2), 239–264 (2005)
23. Kuznetsov, S.O.: On the intractability of computing the Duquenne-Guigues base. Journal of Universal Computer Science **10**(8), 927–933 (2004)

24. Maier, D.: Theory of Relational Databases. Computer Science Pr (1983)
25. Mannila, H., Räihä, K.J.: Algorithms for inferring functional dependencies from relations. Data & Knowledge Engineering **12**(1), 83 – 99 (1994)
26. Obiedkov, S., Duquenne, V.: Attribute-incremental construction of the canonical implication basis. Annals of Mathematics and Artificial Intelligence **49**(1-4), 77–99 (2007)
27. Reppe, H.: Attribute exploration using implications with proper premises. In: P.W. Eklund, O. Haemmerlé (eds.) ICCS, *Lecture Notes in Computer Science*, vol. 5113, pp. 161–174. Springer (2008)
28. Rudolph, S.: Some notes on pseudo-closed sets. In: S.O. Kuznetsov, S. Schmidt (eds.) Proc. of the 5th Int. Conf. on Formal Concept Analysis, *Lecture Notes in Computer Science*, vol. 4390, pp. 151–165. Springer (2007)
29. Ryssel, U., Ploennigs, J., Kabitzsch, K.: Automatic variation-point identification in function-block-based models. In: Proc. of the 9th Int. Conf. on Generative Programming and Component Engineering, pp. 23–32. ACM (2010)
30. Ryssel, U., Ploennigs, J., Kabitzsch, K.: Extraction of feature models from formal contexts. In: Proc. of the 15th Int. Software Product Line Conference, pp. 4:1–4:8. ACM (2011)
31. Schlimmer, J.: UCI Machine Learning Repository: 1984 United Stated congressional voting records (1987). URL `http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`
32. Takata, K.: A worst-case analysis of the sequential method to list the minimal hitting sets of a hypergraph. SIAM J. Discret. Math. **21**(4), 936–946 (2007)
33. Zaki, M.J., Ogihara, M.: Theoretical foundations of association rules. In: Proceedings of the 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (1998)