Fast Computation of Proper Premises

Uwe Ryssel[1], Felix Distel[2], and Daniel Borchmann[3]

[1] Institute of Applied Computer Science, Technische Universität Dresden, Dresden, Germany, uwe.ryssel@tu-dresden.de
[2] Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany, felix@tcs.inf.tu-dresden.de
[3] Institute of Algebra, Technische Universität Dresden, Dresden, Germany, borch@tcs.inf.tu-dresden.de

Abstract. This work is motivated by 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. Since the usual stem base algorithms have proven to be too costly in terms of runtime, we have developed 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 provide heuristic evidence that an approach based on proper premises will also be beneficial for other applications.

1 Introduction

Today, graph-like structures are used in many model languages to specify algorithms or problems in a more readable way. Examples are 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 implicational 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.

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 [10,15], 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 concept intents. Recent theoretical results suggest that existing approaches for computing the stem base may not lead to algorithms with better worst-case complexity [6,1].

Bearing this in mind, we focus on *proper premises*. Just like pseudo-intents, that are used to obtain the stem base, proper premises yield a sound and complete set of implications. Because this set of implications does not have minimal cardinality, proper premises have been outside the focus of the FCA community for many years. However, there are substantial arguments to reconsider using them. Existing methods for computing proper premises avoid computing concept intents. Thus, in contexts with many concept intents they may have a clear advantage in runtime over the stem base algorithms. This is particularly true for our application where 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. 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.

Often, in applications, runtime is the limiting factor, not the size of the basis. 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 [16].

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 conclude by providing experimental results. These show highly significant improvements for the contexts obtained from the model refactoring application. For a sample context, where Next-Closure required several hours 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.

2 Preliminaries

We provide a short summary of the most common definitions in formal concept analysis. 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., $A' = \{ m \in M \mid \forall g \in G\colon gIm \}$. Likewise, for some set $B \subseteq M$ we define $B' = \{ g \in G \mid \forall m \in B\colon gIm \}$. Pairs of the form (A, B) where $A' = B$ and $B' = A$ are called formal concepts. Formal concepts of the form $(\{ m \}', \{ m \}'')$ for some attribute $m \in M$ are called *attribute concept* 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*.

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 $B_1 \to B_2$. 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* $\mathcal{L}$ if for every context $\mathbb{K}$ in which all implications from $\mathcal{L}$ hold, $B_1 \to B_2$ also holds. We say that $\mathcal{L}$ is *sound* for $\mathbb{K}$ if all implications from $\mathcal{L}$ hold in $\mathbb{K}$, and we say that $\mathcal{L}$ is *complete* for $\mathbb{K}$ if all implications that hold in $\mathbb{K}$ follow from $\mathcal{L}$. There exists a sound and complete set of implications for each context which has minimal cardinality [12]. This is called the stem base. The exact definition of the stem base is outside the scope of this work.

A sound and complete set of implications can also 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.,

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

B is called a *proper premise* if $B^\bullet$ is not empty. It is called a *proper premise for* $m \in M$ if $m \in B^\bullet$. It can be shown that $\mathcal{L} = \{ B \to B^\bullet \mid B \text{ proper premise} \}$ is sound and complete [11]. Several alternative ways to define this sound and complete set of implications can be found in [2].

We write $g \swarrow m$ if g' is maximal with respect to the subset order among all object intents which do not contain m.

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 [14]. 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 [11]).

Let V be a finite set of vertices. Then a *hypergraph on* V is simply a pair (V, H) where H is a subset of the power set 2^V. Intuitively, each set $E \in H$ 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 H if it intersects every edge $E \in H$, i.e.,

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

S is called a *minimal hypergraph transversal of* H if it is minimal with respect to the subset order among all hypergraph transversals of H. The *transversal hypergraph* of H is the set of all minimal hypergraph transversals of H. It is denoted by $Tr(H)$. The problem of deciding for two hypergraphs G and H whether H is the transversal hypergraph of G is called TransHyp. The problem of enumerating all minimal hypergraph transversals of a hypergraph 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 quasipolynomial time [9], it is not believed to be coNP-complete. Furthermore, it has been shown that it can be decided using only limited non-determinism [8]. 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 [8,4].

The following proposition can be found in [11] among others.

Proposition 1. *$P \subseteq M$ is a premise of $m \in M$ iff*

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

holds for all $g \in G$ with $g \swarrow m$. 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 (M, H) where*

$$H := \{M \setminus g' \mid g \in G, g \swarrow m\}.$$

The set of all proper premises of m is exactly the transversal hypergraph

$$Tr(\{M \setminus g' \mid g \in G, g \swarrow m\}).$$

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 [14]. 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 [2] and the enumeration of set covers.

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 of more than one attribute. Our goal is to avoid this wherever possible.

The first idea is shown in Algorithm 2. There we introduce a *candidate set* C of particular attributes, depending on the current attribute m. We claim now that we only have to search for minimal hypergraph transversals P of $\{ M \setminus g' \mid g \not\swarrow m \}$ with $P \subseteq C$. We provide some intuition for this idea.

Algorithm 1 Naïve Algorithm for Enumerating All Proper Premises

```
**Input:** $\mathbb{K} = (G, M, I)$
$\mathcal{P} = \emptyset$
**for all** $m \in M$ **do**
    $\mathcal{P} = \mathcal{P} \cup Tr(\{M \setminus g' \mid g \in G, g \not\swarrow m\})$
**end for**
**return** $\mathcal{P}$
```

Algorithm 2 A Better Algorithm for Enumerating All Proper Premises

```
**Input:** $\mathbb{K} = (G, M, I)$
$\mathcal{P} = \{ \{ m \} \mid m \in M, \{ m \}$ is a proper premise of $\mathbb{K} \}$
**for all** $m \in M$ **do**
    $C = \{ u \in M \setminus \{ m \} \mid \nexists v \in M : \mu u \wedge \mu m \leq \mu v < \mu m \}$
    $\mathcal{P} = \mathcal{P} \cup \{ P \subseteq C \mid P$ minimal hypergraph transversal of $\{ M \setminus g' \mid g \not\swarrow m \} \}$
**end for**
**return** $\mathcal{P}$
```

Let us fix a formal context $\mathbb{K} = (G, M, I)$, choose $m \in M$ and let $P \subseteq M$ be a proper premise for m. Then we know that $m \in P''$, which is equivalent to

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

If we now find another attribute $n \in M \setminus \{ m \}$ with

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

it suffices to find the set P as a proper premise for n, because from $\mu n < \mu m$ we can already infer $m \in P''$. Conversely, if we search for all proper premises for m, we only have to search for those who are not proper premises for attributes n with $\mu n < \mu m$. Now suppose that there exists an element $u \in P$ and an attribute $v \in M$ such that

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

Then we know

$$\Big(\bigwedge_{p \in P} \mu p\Big) \wedge \mu m = \bigwedge_{p \in P} \mu p \leq \mu v < \mu m,$$

i.e., P is already a proper premise for v. In this case, we do not have to search for P, since it will be found in another iteration. On the other hand, if P is a proper premise for m but not for any other attribute $n \in M$ with $\mu n < \mu m$, the argument given above shows that an element $u \in P$ and an attribute $v \in M$ satisfying (1) cannot exist.

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 $P = \{ c \}$ 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 $P \subseteq C$. 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 $p \in C$ and all together $P \subseteq C$.

To complete the proof it remains to show that P is a minimal hypergraph transversal of $\{ M \setminus \{ g \}' \mid g \swarrow c \}$, 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.

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., $\mu m = \bigwedge_{i=1}^{n} \mu x_i$. 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.

Algorithm 3 Computing Enough Proper Premises

```
Input: $\mathbb{K} = (G, M, I)$
$\mathcal{P} = \{\, \{\, m \,\} \mid m \in M, \{\, m \,\}$ is a proper premise of $\mathbb{K} \,\}$
$N = M \setminus \{\, x \in M \mid \mu x = \bigwedge_{i=1}^{n} \mu x_i$ for an $n \in \mathbb{N}$ and $x_i \in M$ for $1 \leq i \leq n \,\}$
for all $m \in M$ do
  $C = \{\, u \in N \setminus \{\, m \,\} \mid \nexists v \in M : \mu u \wedge \mu m \leq \mu v < \mu m \,\}$
  $\mathcal{P} = \mathcal{P} \cup \{\, P \subseteq C \mid P$ minimal hypergraph transversal of $\{\, M \setminus g' \mid g \not\swarrow m \,\} \,\}$
end for
return $\mathcal{P}$
```

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*

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

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

Proof. Let m be an attribute and let P be a proper premise for m. If $P \notin \mathcal{P}$ 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 $\{ y_1 \} \in \mathcal{P}$, since all singleton proper premises are contained in $\mathcal{P}$. If $P_1 \notin \mathcal{P}$ 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 $\mathcal{P}' = \{ \{ y_1 \}, \ldots, \{ y_k \}, P_k \} \subseteq \mathcal{P}$ such that $P \to \{ m \}$ follows from $\{ Q \to Q^\bullet \mid Q \in \mathcal{P}' \}$. Thus $\mathcal{L}$ is a sound and complete set of implications.

Together with Lemma 1 this yields correctness of Algorithm 3.

Corollary 2. *The set of proper premises computed by Algorithm 3 yields a sound and complete set of implications for the given formal context.*

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. In the classical case the reason is that the number of intents can be exponential in the size of the stem base [13]. 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 1 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 $\{m_i \mid i \in I\} \cup \{m'_i \mid i \notin I\}$ for some $I \subseteq \{1, \dots, n\}$, while the only pseudo-intents are the singleton sets and $\{m_1, \dots, m_n, m'_1, \dots, m'_n\}$. Hence there are 2^n proper premises for a, while there are only $2n + 2$ pseudo-intents.

Table 1. Context $\mathbb{K}_n$ with Exponentially Many Proper Premises

	$m_1 \dots m_n$	$m'_1 \dots m'_n$	a
g_1			
$\vdots$	$I_{\neq}$	$I_{\neq}$	
g_n			

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.[4] Derivations of these formulae can be found in [7].

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

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

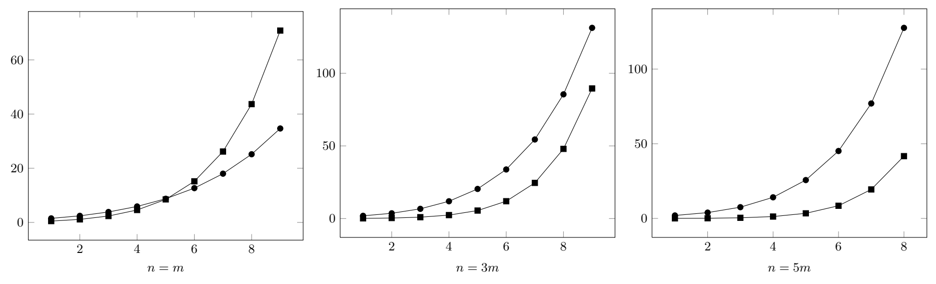
Figure 1 shows the values of $m \cdot \mathbb{E}_{\text{pp}}$ (squares) and $\mathbb{E}_{\text{intent}}$ (bullets) for quadratic contexts and for contexts with $n = 3m$ and $n = 5m$. While there are more proper premises for quadratic contexts, less proper premises need to be computed for contexts with a large number of objects.

5.2 Applications in Model Refactoring

In the Section 5.3 we shall see that the proper premise approach performs extremely well on contexts obtained from model refactoring. We briefly describe how these contexts are obtained. Working with data-flow-oriented simulation

[4] We ignore renaming of attributes and objects.

Fig. 1. Expected Number of Intents and Proper Premises for Certain Families of Formal Contexts



models will result fast in many variants of similar models, which are difficult to manage. One solution for that problem is to refactor these variants to configurable models, containing all parts and information about the valid combinations. Using a generator, the single variants, which are merged, can be created later on demand.

Using matching algorithms, corresponding artifacts (i.e., blocks, connections, and parameter settings) among the variants can be found. This results in a minimal set of artifacts, from that each variant can be built. To restrict the combinations, and with it the number of variants that can be generated, to the given model variants, we have to specify the dependencies among the artifacts. These can be defined by a set of implications, which can be calculated by the formal concept analysis. The whole process is described in [17,18].

Using the matching result, a formal context can be built: The model variants form the object set and the artifacts form the attribute set. The relation between variants and artifacts is defined by incidence. If a block or connection exists in a variant, they are related. The many-valued relation among variants and parameters is resolved by using parameter settings (i.e., pairs of parameters and their values) as attributes. The resulting implications should also contain negated literals, so we also need the negated counterparts of the artifact attributes. The semantic of the negated attributes is in this case a non-existing block, a non-existing connection, and a parameter that is not set to a certain value. In our application, only negated counterparts of irreducible attributes are needed.

This results in dense contexts (usually with a fill ratio greater than 0.3) with a typical size of 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 name 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 Next-Closure, *HT* which computes all proper premises as hypergraph transversals as in Algorithm 1, and *PP*, an implementation of Algorithm 3.

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 [3]. 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 [5], 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 [19]. 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.[5] The third structured data set originates from the application described in Section 5.2 and [18]. It has 26 objects, 79 attributes and an approximate density of 0.35.

The other part of testing data sets consist 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 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 2 and Table 3.

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

[5] Note that this is another scaling as the one in [15], so the times obtained there cannot be compared directly to ours.

Table 2. Behaviour on Structured Data

Context	SB		HT		PP	
Data-Flow-Model	6.5 hrs	86	37 hrs	1480	0.1 sec	86
SPECT	175 sec	1778	16 sec	6830	5.4 sec	6830
Voting	13 hrs	18,572	6 hrs	140,032	17 min	140,032

Table 3. Behaviour on Random Data

Context	SB		HT		PP	
$20 \times 40 \times 0.9$	75 sec	78	0.8 sec	988	1.4 sec	527
$20 \times 40 \times 0.8$	820 sec	879	4.3 sec	11263	2.4 sec	9223
$20 \times 40 \times 0.3$	8.9 sec	556	4.6 sec	3780	2.5 sec	3698
$20 \times 40 \times 0.2$	5.2 sec	386	2.2 sec	1817	0.9 sec	1478
$40 \times 20 \times 0.9$	17 sec	62	0.04 sec	105	0.04 sec	105
$40 \times 20 \times 0.8$	104 sec	920	1.5 sec	2017	0.45 sec	2017
$40 \times 20 \times 0.3$	1.6 sec	388	1 sec	1258	0.7 sec	1258
$40 \times 20 \times 0.2$	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	2533	0.4 sec	2533
$25 \times 25 \times 0.3$	1.2 sec	252	0.8 sec	1231	0.6 sec	1231
$25 \times 25 \times 0.2$	0.9 sec	226	0.34 sec	550	0.3 sec	533

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 SB and HT/PP, 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 tradeoff 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.

References

1. Mikhail Babin and Sergei Kuznetsov. Recognizing pseudo-intents is coNP-complete. In Marzena Kryszkiewicz and Sergei Obiedkov, editors, *Proc. of the*

7th Int. Conf. on Concept Lattices and Their Applications, volume 672. CEUR Workshop Proceedings, 2010.

2. K. Bertet and B. Monjardet. The multiple facets of the canonical direct unit implicational basis. *Theoretical Computer Science*, 411(22–24):2155–2166, 2010.
3. Daniel Borchmann. conexp-clj. `http://www.math.tu-dresden.de/~borch/conexp-clj/`.
4. E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension. *Parallel Processing Letters*, 10:253–266, 2000.
5. Krzysztof J. Cios, Lukasz A. Kurgan, and Lucy S. Goodenday. UCI Machine Learning Repository: SPECT Heart Data Set, 2001.
6. Felix Distel. Hardness of enumerating pseudo-intents in the lectic order. In Barış Sertkaya and Léonard Kwuida, editors, *Proc. of the 8th Int. Conf. on Formal Concept Analysis*, volume 5986 of *Lecture Notes in Artificial Intelligence*, pages 124–137. Springer, 2010.
7. Felix Distel and Daniel Borchmann. Expected numbers of proper premises and concept intents. Preprint, Institut für Algebra, TU Dresden, 2011.
8. Thomas Eiter, Georg Gottlob, and Kazuhisa Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM J. Comput.*, 32:514–537, February 2003.
9. Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618 – 628, 1996.
10. Bernhard Ganter. Two basic algorithms in concept analysis. Preprint 831, Fachbereich Mathematik, TU Darmstadt, Darmstadt, Germany, 1984.
11. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, New York, 1997.
12. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Humaines*, 95:5–18, 1986.
13. Sergei O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *Journal of Universal Computer Science*, 10(8):927–933, 2004.
14. Heikki Mannila and Kari-Jouko Räihä. Algorithms for inferring functional dependencies from relations. *Data & Knowledge Engineering*, 12(1):83 – 99, 1994.
15. Sergei Obiedkov and Vincent Duquenne. Attribute-incremental construction of the canonical implication basis. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):77–99, 2007.
16. Sebastian Rudolph. Some notes on pseudo-closed sets. In Sergei O. Kuznetsov and Stefan Schmidt, editors, *Proc. of the 5th Int. Conf. on Formal Concept Analysis*, volume 4390 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2007.
17. Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. Automatic variation-point identification in function-block-based models. In *Proc. of the 9th Int. Conf. on Generative Programming and Component Engineering*, pages 23–32. ACM, 2010.
18. Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. Extraction of feature models from formal contexts. In *Proc. of the 15th Int. Software Product Line Conference*, pages 4:1–4:8. ACM, 2011.
19. Jeff Schlimmer. UCI Machine Learning Repository: 1984 United Stated congressional voting records, 1987.