NextClosures: Parallel Computation of the Canonical Base

Francesco Kriegel and Daniel Borchmann

Institute of Theoretical Computer Science, TU Dresden, Germany
{francesco.kriegel,daniel.borchmann}@tu-dresden.de

Abstract. The canonical base of a formal context plays a distinguished role in formal concept analysis. This is because it is the only minimal base so far that can be described explicitly. For the computation of this base several algorithms have been proposed. However, all those algorithms work sequentially, by computing only one pseudo-intent at a time – a fact which heavily impairs the practicability of using the canonical base in real-world applications. In this paper we shall introduce an approach that remedies this deficit by allowing the canonical base to be computed in a parallel manner. First experimental evaluations show that for sufficiently large data-sets the speedup is proportional to the number of available CPUs.

Keywords: Formal Concept Analysis, Canonical Base, Parallel Algorithms

1 Introduction

The implicational theory of a formal context is of interest in a large variety of applications. In those cases, computing the canonical base of the given context is often desirable, as it has minimal cardinality among all possible bases. On the other hand, conducting this computation often imposes a major challenge, often endangering the practicability of the underlying approach.

There are two known algorithms for computing the canonical base of a formal context [6, 12]. Both algorithms work sequentially, i.e. they compute one implication after the other. Moreover, both algorithms compute in addition to the implications of the canonical base all formal concepts of the given context. This is a disadvantage, as the number of formal concepts can be exponential in the size of the canonical base. On the other hand, the size of the canonical base can be exponential in the size of the underlying context [10]. Additionally, up to today it is not known whether the canonical base can be computed in output-polynomial time, and certain complexity results hint at a negative answer [3]. For the algorithm from [6], and indeed for any algorithm that computes the pseudo-intents in a lectic order, it has been shown that it cannot compute the canonical base with polynomial delay [2].

However, the impact of theoretical complexity results for practical application is often hard to access, and it is often worth investigating faster algorithm for theoretically intractable results. A popular approach is to explore the possibilities to parallelize known sequential algorithms. This is also true for formal concept analysis, as can be seen in the development of parallel versions for computing the concept lattice of a formal context [5, 13].

In this work we want to investigate the development of a parallel algorithm for computing the canonical base of a formal context $\mathbb{K}$. The underlying idea is actually

quite simple, and has been used by Lindig [11] to (sequentially) compute the concept lattice of a formal context: to compute the canonical base, we compute the lattice of all *intents* and *pseudo-intents* of $\mathbb{K}$. This lattice can be computed bottom up, in a level-wise order, and this computation can be done in parallel provided that the lattice has a certain "width" at a particular level. The crucial fact now is that the upper neighbours of an intent or pseudo-intent B can be easily computed by just iterating over all attributes $m \notin B$ and computing the closure of $B \cup \{ m \}$. In the approach of Lindig mentioned above this closure is just the usual double-prime operation $B \mapsto B^{II}$ of the underlying formal context $\mathbb{K}$. In our approach it is the closure operator whose closures are exactly the intents and pseudo-intents of $\mathbb{K}$. Surprisingly, despite the simpleness of our approach, we are not aware of any prior work on computing the canonical base of a formal context in a parallel manner. Furthermore, experimental results presented in this work indicate that for suitable large data-sets the computation of the canonical base can be speed up by a factor proportional to the number of available CPUs.

The paper is structured as follows. After recalling all necessary notions of formal concept analysis in Section 2, we shall describe in Section 3 our approach of computing the canonical base in parallel. Benchmarks of this algorithm are presented in Section 4, and we shall close this work with some conclusions in Section 5.

2 Preliminaries

This section gives a brief overview on the notions of formal concept analysis [7] that are used in this document. The basic structure is a *formal context* $\mathbb{K} = (G, M, I)$ consisting of a set G of *objects*, a set M of *attributes*, and an *incidence relation* $I \subseteq G \times M$. For a pair $(g, m) \in I$ we also use the infix notation $g\ I\ m$, and say that the object g *has* the attribute m. Each formal context $\mathbb{K}$ induces the *derivation operators* $\cdot^I \colon \mathfrak{P}(G) \to \mathfrak{P}(M)$ and $\cdot^I \colon \mathfrak{P}(M) \to \mathfrak{P}(G)$ that are defined as follows for object sets $A \subseteq G$ and attribute sets $B \subseteq M$:

$$A^I \coloneqq \{ m \in M \mid \forall g \in A \colon (g, m) \in I \} \text{ and } B^I \coloneqq \{ m \in M \mid \forall m \in B \colon (g, m) \in I \}.$$

In other words, A^I is the set of all attributes that all objects from A have in common, and dually B^I is the set of all objects which have all attributes from B. A *formal concept* of $\mathbb{K}$ is a pair (A, B) such that $A^I = B$ and $B = A^I$, and the set of all formal concepts of $\mathbb{K}$ is denoted by $\mathfrak{B}(\mathbb{K})$.

An *implication* over the set M is an expression of the form $X \to Y$ where $X, Y \subseteq M$. An implication $X \to Y$ over M is *valid* in $\mathbb{K}$ if $X^I \subseteq Y^I$. A set $\mathcal{L}$ of implications over M is *valid* in $\mathbb{K}$ if each implication in $\mathcal{L}$ is valid in $\mathbb{K}$. An implication $X \to Y$ *follows* from the set $\mathcal{L}$ if $X \to Y$ is valid in every context with attribute set M in which $\mathcal{L}$ is valid. Furthermore, a *model* of $X \to Y$ is a set $T \subseteq M$ such that $X \subseteq T$ implies $Y \subseteq T$. A *model* of $\mathcal{L}$ is a model of all implications in $\mathcal{L}$, and $X^{\mathcal{L}}$ is the smallest superset of X that is a model of $\mathcal{L}$. The set $X^{\mathcal{L}}$ can be computed as follows.

$$X^{\mathcal{L}} \coloneqq \bigcup\nolimits_{n \geq 1} X^{\mathcal{L}_n} \quad \text{where} \quad X^{\mathcal{L}_1} \coloneqq X \cup \bigcup \{ B \mid A \to B \in \mathcal{L} \text{ and } A \subseteq X \}$$
$$\text{and} \quad X^{\mathcal{L}_{n+1}} \coloneqq (X^{\mathcal{L}_n})^{\mathcal{L}_1} \quad \text{for all } n \in \mathbb{N}.$$

The following lemma shows some well-known equivalent statements for entailment of implications from implication sets. We will not prove them here.

Lemma 1. *Let $\mathcal{L} \cup \{ X \to Y \}$ be a set of implications over M. Then the following statements are equivalent:*

1. *$X \to Y$ follows from $\mathcal{L}$.*
2. *If $\mathbb{K}$ is a formal context with attribute set M such that $\mathcal{L}$ is valid in $\mathbb{K}$, then $X \to Y$ is also valid in $\mathbb{K}$.*
3. *If $T \subseteq M$ and T is a model of $\mathcal{L}$, then T is a model of $X \to Y$.*
4. *$Y \subseteq X^{\mathcal{L}}$.*

An attribute set $B \subseteq M$ is called *intent* of $\mathbb{K} = (G, M, I)$ if $B = B^{II}$. An attribute set $P \subseteq M$ is called *pseudo-intent* of $\mathbb{K}$ if $P \neq P^{II}$, and furthermore for each pseudo-intent $Q \subsetneq P$ the set inclusion $Q^{II} \subseteq P$ is satisfied. We denote the set of all pseudo-intents of $\mathbb{K}$ by $\mathrm{PsInt}(\mathbb{K})$. Then the *canonical implicational base* of $\mathbb{K}$ is defined as the following implication set:

$$\{ P \to P^{II} \mid P \in \mathrm{PsInt}(\mathbb{K}) \}.$$

The canonical base has the property that it is a *minimal base* of $\mathbb{K}$, i.e. it is a *base* of $\mathbb{K}$, meaning that it is a set of valid implications of $\mathbb{K}$ such that every valid implication of $\mathbb{K}$ is entailed by it. Furthermore, its cardinality is minimal among all bases of $\mathbb{K}$.

It is readily verified that a subset $X \subseteq M$ is an intent or a pseudo-intent of $\mathbb{K}$ if and only if X is a closure of the closure operator $\mathbb{K}^*$ that is defined as follows:

$$X^{\mathbb{K}^*} := \bigcup\nolimits_{n \geq 1} X^{\mathbb{K}^*_n} \quad \text{where} \quad X^{\mathbb{K}^*_1} := X \cup \bigcup \{ P^{II} \mid P \in \mathrm{PsInt}(\mathbb{K}) \text{ and } P \subsetneq X \}$$
$$\text{and} \quad X^{\mathbb{K}^*_{n+1}} := (X^{\mathbb{K}^*_n})^{\mathbb{K}^*_1} \quad \text{for all } n \in \mathbb{N}.$$

Of course, if $\mathcal{L}$ is the canonical base of $\mathbb{K}$ as described above, then both closure operators $\mathbb{K}^*$ and $\mathcal{L}^*$ coincide, where $\mathcal{L}^*$ is defined by the following equations:

$$X^{\mathcal{L}^*} := \bigcup\nolimits_{n \geq 1} X^{\mathcal{L}^*_n} \quad \text{where} \quad X^{\mathcal{L}^*_1} := X \cup \bigcup \{ B \mid A \to B \in \mathcal{L} \text{ and } A \subsetneq X \}$$
$$\text{and} \quad X^{\mathcal{L}^*_{n+1}} := (X^{\mathcal{L}^*_n})^{\mathcal{L}^*_1} \quad \text{for all } n \in \mathbb{N}.$$

3 Parallel Computation of the Canonical Base

The well-known `NextClosure` algorithm developed by Ganter [6] can be used to enumerate the implications of the canonical base. The mathematical idea behind this algorithm is to compute all intents and pseudo-intents of our formal context $\mathbb{K}$ in a certain linear order, namely the *lectic order*. As an advantage the next (pseudo-)intent is uniquely determined, but we potentially have to do backtracking in order to find it. It can be seen quite easily that those sets form a complete lattice, and the `NextClosure` algorithm uses the closure operator $\mathbb{K}^*$ of this lattice to enumerate the pseudo-intents of $\mathbb{K}$ in the lectic order. Furthermore, this algorithm is inherently sequential, i.e. it is not possible to parallelize it.

In our approach we shall not make use of the lectic order. Indeed, our algorithm will enumerate all intents and pseudo-intents of $\mathbb{K}$ in the subset-order, with no further restrictions. As a benefit we get a very easy and obvious way to parallelize this enumeration. Moreover, in multi-threaded implementations no communication between different threads is necessary. However, as it is the case with all other known algorithms

for computing the canonical base, we also have to compute all intents in addition to all pseudo-intents of the given formal context.

The main idea is very simple and works as follows. From the definition of pseudo-intents we see that in order to decide whether an attribute set $P \subseteq M$ is a pseudo-intent we only need all pseudo-intents $Q \subsetneq P$, i.e. it suffices to know all pseudo-intents with a smaller cardinality than P. This allows for the level-wise computation of the canonical base w.r.t. the subset inclusion order, i.e. we can enumerate the (pseudo-)intents w.r.t. increasing cardinality.

An algorithm that implements this idea works as follows. First we start by considering the empty set, as it is the only set with cardinality 0. Of course, the empty set must either be an intent or a pseudo-intent, and the distinction can be made by checking whether $\emptyset = \emptyset^{II}$. Then assuming inductively that all pseudo-intents with cardinality $< k$ have been determined, we can correctly decide whether a subset $P \subseteq M$ with $|P| = k$ is a pseudo-intent or not.

To compute the lattice of intents and pseudo-intents of $\mathbb{K}$ the algorithm manages a set of *candidates* that contains the (pseudo-)intents on the current level. Then, whenever a pseudo-intent P has been found, the $\subseteq$-next closure is uniquely determined by its closure P^{II} in the context $\mathbb{K}$. If an intent B has been found, then the $\subseteq$-next closures must be of the form $(B \cup \{ m \})^{\mathbb{K}^*}$, $m \notin B$. However, as we are not aware of the full implicational base of $\mathbb{K}$ yet, but only of an *approximation* $\mathcal{L}$ of it, the operators $\mathbb{K}^*$ and $\mathcal{L}^*$ do not coincide on all subsets of M. We will show that they yield the same closure for attribute sets $B \subseteq M$ with a cardinality $|B| \leq k$ if $\mathcal{L}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with cardinality $|P| < k$. Consequently, the $\mathcal{L}^*$-closure of a set $B \cup \{ m \}$ may not be an intent or pseudo-intent of $\mathbb{K}$. Instead they are added to the candidate list, and are processed when all pseudo-intents with smaller cardinality have been determined. We will formally prove that this technique is correct. Furthermore, the computation of all pseudo-intents and intents of cardinality k can be done in parallel, since they are independent of each other.

In summary, we shortly describe the inductive structure of the algorithm as follows: Let $\mathbb{K}$ be a finite formal context. We use four variables: k denotes the current cardinality of candidates, $\mathbf{C}$ is the set of candidates, $\mathfrak{B}$ is a set of formal concepts, and $\mathcal{L}$ is an implication set. Then the algorithm works as follows.

1. Set $k \coloneqq 0$, $\mathbf{C} \coloneqq \{ \emptyset \}$, $\mathfrak{B} \coloneqq \emptyset$, and $\mathcal{L} \coloneqq \emptyset$.
2. In parallel: For each candidate set $C \in \mathbf{C}$ with cardinality $|C| = k$ determine whether it is $\mathcal{L}^*$-closed. If not, then add its $\mathcal{L}^*$-closure to the candidate set $\mathbf{C}$, and go to Step 5.
3. If C is an intent of $\mathbb{K}$, then add the formal concept (C^I, C) to $\mathfrak{B}$. Otherwise C must be a pseudo-intent, and thus we add the formal implication $C \to C^{II}$ to the set $\mathcal{L}$, and add the formal concept (C^I, C^{II}) to the set $\mathfrak{B}$.
4. For each observed intent C^{II}, add all its upper neighbours $C^{II} \cup \{ m \}$ where $m \notin C^{II}$ to the candidate set $\mathbf{C}$.
5. Wait until all candidates of cardinality k have been processed. If $k < |M|$, then increase the candidate cardinality k by 1, and go to Step 2. Otherwise return $\mathfrak{B}$ and $\mathcal{L}$.

In order to approximate the operator $\mathcal{L}^*$ we furthermore introduce the following notion: If $\mathcal{L}$ is a set of implications, then $\mathcal{L}\restriction_k$ denotes the subset of $\mathcal{L}$ that consists of all implications whose premises have a cardinality of at most k.

Lemma 2. *Let $\mathbb{K} = (G, M, I)$ be a formal context, $\mathcal{L}$ its canonical implicational base, and $X \subseteq M$ an attribute set. Then the following statements are equivalent:*

1. *X is either an intent or a pseudo-intent of $\mathbb{K}$.*
2. *X is $\mathbb{K}^*$-closed.*
3. *X is $\mathcal{L}^*$-closed.*
4. *X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed.*
5. *There is a $k \geq |X| - 1$ such that X is $(\mathcal{L}\restriction_k)^*$-closed.*
6. *For all $k \geq |X| - 1$ it holds that X is $(\mathcal{L}\restriction_k)^*$-closed.*

Proof. 1⇔2. If X is an intent or a pseudo-intent, then it is obviously $\mathbb{K}_1^*$-closed, i.e. $\mathbb{K}^*$-closed. Vice versa, if X is $\mathbb{K}^*$-closed, but no intent, then X contains the closure P^{II} of every pseudo-intent $P \subsetneq X$, and hence X must be a pseudo-intent. 2⇔3. is obvious. 3⇔4. follows directly from the fact that $P \subsetneq X$ implies $|P| < |X|$. 4⇔5. The only-if-direction is trivial. Consider $k \geq |X| - 1$ such that X is $(\mathcal{L}\restriction_k)^*$-closed. Then X contains all conclusions B where $A \to B \in \mathcal{L}$ is an implication with premise $A \subsetneq X$ such that $|A| \leq k$. Of course, $A \subsetneq X$ implies $|A| < |X|$, and thus X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed as well. 4⇔6. The only-if-direction is trivial. Finally, assume that $k \geq |X| - 1$ and X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed. Obviously, there are no subsets $A \subsetneq X$ with $|X| \leq |A| \leq k$, and so X must be $(\mathcal{L}\restriction_k)^*$-closed, too. □

As an immediate consequence of Lemma 2 we infer that in order to decide the $\mathbb{K}^*$-closedness of an attribute set X it suffices to know all implications in the canonical base whose premise has a lower cardinality than X.

Corollary 3. *If $\mathcal{L}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with $|P| < k$, and otherwise only implications with premise cardinality k, then for all attribute sets $X \subseteq M$ with $|X| \leq k$ the following statements are equivalent:*

1. *X is an intent or a pseudo-intent of $\mathbb{K}$.*
2. *X is $\mathcal{L}^*$-closed.*

This corollary allows us in a certain sense to approximate the set of all $\mathbb{K}^*$-closures w.r.t. increasing cardinality, and thus also permits the approximation of the closure operator $\mathcal{L}^*$ where $\mathcal{L}$ is the canonical base of $\mathbb{K}$. In the following Lemma 4 we will characterise the structure of the lattice of all $\mathbb{K}^*$-closures, and also give a method to compute upper neighbours. It is true that between comparable pseudo-intents there must always be an intent. In particular, the unique upper $\mathbb{K}^*$-closed neighbour of a pseudo-intent must be an intent.

Lemma 4. *Let $\mathbb{K}$ be a formal context. Then the following statements are true:*

1. *If $P \subseteq M$ is a pseudo-intent, then there is no intent or pseudo-intent strictly between P and P^{II}.*
2. *If $B \subseteq M$ is an intent, then the next intents or pseudo-intents are of the form $(B \cup \{ m \})^{\mathbb{K}^*}$ for attributes $m \notin B$.*
3. *If $X \subsetneq Y \subseteq M$ are neighbouring $\mathbb{K}^*$-closures, then $Y = (X \cup \{ m \})^{\mathbb{K}^*}$ for all attributes $m \in Y \setminus X$.*

Algorithm 1 `NextClosures` $(\mathbb{K})$

```
1  k := 0, C := { ∅ }, B := ∅, L := ∅
2  while k ≤ |M| do
3      for all C ∈ C with |C| = k do in parallel
4          C := C \ { C }
5          if C = C^{L*} then
6              if C ≠ C^{II} then
7                  L := L ∪ { C → C^{II} }
8              B := B ∪ { (C^I, C^{II}) }
9              C := C ∪ { C^{II} ∪ { m } | m ∉ C^{II} }
10         else
11             C := C ∪ { C^{L*} }
12     Wait for termination of all parallel processes.
13     k := k + 1
14 return (B, L)
```

Proof. 1. Let $P \subseteq M$ be a pseudo-intent of $\mathbb{K}$. Then for every intent B between P and P^{II}, i.e. $P \subseteq B \subseteq P^{II}$, we have $B = B^{II} = P^{II}$. Thus, there cannot be an intent strictly between P and P^{II}. Furthermore, if Q were a pseudo-intent such that $P \subsetneq Q \subseteq P^{II}$, then $P^{II} \subseteq Q$, and thus $Q = P^{II}$, a contradiction.

2. Let $B \subseteq M$ be an intent of $\mathbb{K}$, and $X \supseteq B$ an intent or pseudo-intent of $\mathbb{K}$ such that there is no other intent or pseudo-intent between them. Then $B \subseteq B \cup \{ m \} \subseteq X$ for every $m \in X \setminus B$. Thus, $B = B^{\mathbb{K}^*} \subsetneq (B \cup \{ m \})^{\mathbb{K}^*} \subseteq X^{\mathbb{K}^*} = X$. Then $(B \cup \{ m \})^{\mathbb{K}^*}$ is an intent or a pseudo-intent between B and X that strictly contains B, and hence $X = (B \cup \{ m \})^{\mathbb{K}^*}$.

3. Consider an attribute $m \in Y \setminus X$. Then $X \cup \{ m \} \subseteq Y$, and thus $X \subsetneq (X \cup \{ m \})^{\mathbb{K}^*} \subseteq Y$, as Y is already closed. Therefore, $(X \cup \{ m \})^{\mathbb{K}^*} = Y$. □

We are now ready to formulate our algorithm `NextClosures` in pseudo-code, see Algorithm 1. In the remainder of this section we shall show that this algorithm always terminates for finite formal contexts $\mathbb{K}$, and that it returns the canonical base as well as the set of all formal concepts of $\mathbb{K}$. Beforehand, let us introduce the following notation:

1. `NextClosures` is *in state* k if it has processed all candidate sets with a cardinality $\leq k$, but none of cardinality $> k$.
2. $\mathbf{C}_k$ denotes the set of candidates in state k.
3. $\mathcal{L}_k$ denotes the set of implications in state k.
4. $\mathfrak{B}_k$ denotes the set of formal concepts in state k.

Proposition 5. *Let $\mathbb{K}$ be a formal context, and assume that* `NextClosures` *has been started on $\mathbb{K}$ and is in state k. Then the following statements are true:*

1. *$\mathbf{C}_k$ contains all pseudo-intents of $\mathbb{K}$ with cardinality $k+1$, and all intents of $\mathbb{K}$ with cardinality $k+1$ whose corresponding formal concept is not already in $\mathfrak{B}_k$.*
2. *$\mathfrak{B}_k$ contains all formal concepts of $\mathbb{K}$ whose intent has cardinality $\leq k$.*
3. *$\mathcal{L}_k$ contains all implications $P \to P^{II}$ where the premise P is a pseudo-intent of $\mathbb{K}$ with cardinality $\leq k$.*
4. *Between the states k and $k+1$ an attribute set with cardinality $k+1$ is an intent or pseudo-intent of $\mathbb{K}$ if and only if it is $\mathcal{L}^*$-closed.*

Proof. We prove the statements by induction on k. The base case handles the initial state $k = -1$. Of course, $\emptyset$ is always an intent or a pseudo-intent of $\mathbb{K}$. Furthermore, it is the only attribute set of cardinality 0 and contained in the candidate set $\mathbf{C}$. As there are no sets with cardinality ≤ -1, $\mathfrak{B}_{-1}$ and $\mathcal{L}_{-1}$ trivially satisfy Statements 2 and 3. Finally, we have that $\mathcal{L}_{-1} = \emptyset$, and hence every attribute set is $\mathcal{L}^*_{-1}$-closed, in particular $\emptyset$.

We now assume that the induction hypothesis is true for k. For every implication set $\mathcal{L}$ between states k and $k+1$, i.e. $\mathcal{L}_k \subseteq \mathcal{L} \subseteq \mathcal{L}_{k+1}$, the induction hypothesis yields that $\mathcal{L}$ contains all formal implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with cardinality $\leq k$, and furthermore only implications whose premises have cardinality $k+1$ (by definition of Algorithm 1). Additionally, we know that the candidate set $\mathbf{C}$ contains all pseudo-intents P of $\mathbb{K}$ where $|P| = k+1$, and all intents B of $\mathbb{K}$ such that $|B| = k+1$ and $(B^I, B) \notin \mathfrak{B}$. Corollary 3 immediately yields the validity of Statements 2 and 3 for $k+1$, as those $\mathbb{K}^*$-closures are recognized correctly in line 5. Then $\mathcal{L}_{k+1}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with $|P| \leq k+1$, and hence each implication set $\mathcal{L}$ with $\mathcal{L}_{k+1} \subseteq \mathcal{L} \subseteq \mathcal{L}_{k+2}$ contains all those implications and furthermore only implications with a premise cardinality $k+2$. By another application of Corollary 3 we conclude that also Statement 4 is satisfied for $k+1$.

Finally, we show Statement 1 for $k+1$. Consider any $\mathbb{K}^*$-closed set X where $|X| = k+2$. Then Lemma 4 states that for all lower $\mathbb{K}^*$-neighbours Y and all $m \in X \setminus Y$ it is true that $(Y \cup \{ m \})^{\mathbb{K}^*} = X$. We proceed with a case distinction.

If there is a lower $\mathbb{K}^*$-neighbour Y which is a pseudo-intent, then Lemma 4 yields that the (unique) next $\mathbb{K}^*$-neighbour is obtained as Y^{II}, and the formal concept (Y^I, Y^{II}) is added to the set $\mathfrak{B}$ in line 8. Of course, it is true that $X = Y^{II}$.

Otherwise all lower $\mathbb{K}^*$-neighbours Y are intents, and in particular this is the case for X being a pseudo-intent by Lemma 4. Then for all these Y we have $(Y \cup \{ m \})^{\mathbb{K}^*} = X$ for all $m \in X \setminus Y$. Furthermore, all sets Z with $Y \cup \{ m \} \subsetneq Z \subsetneq X$ are not $\mathbb{K}^*$-closed. Since $X \setminus Y$ is finite, the following sequence must also be finite:

$$C_0 \coloneqq Y \cup \{ m \} \text{ and } C_{i+1} \coloneqq C_i^{\mathcal{L}^*} \text{ where } \mathcal{L}_{|C_i|-1} \subseteq \mathcal{L} \subseteq \mathcal{L}_{|C_i|}.$$

The sequence is well-defined, since implications from $\mathcal{L}_{|C_i|} \setminus \mathcal{L}_{|C_i|-1}$ have no influence on the closure of C_i. Furthermore, the sequence obviously ends with the set X, and contains no further $\mathbb{K}^*$-closed sets, and each of the sets $C_0, C_1, \ldots$ appears as a candidate during the run of the algorithm, cf. lines 9 and 11. □

From the previous result we can infer that in the last state $|M|$ the set $\mathfrak{B}$ contains all formal concepts of the input context $\mathbb{K}$, and that $\mathcal{L}$ is the canonical base of $\mathbb{K}$. Both sets are returned from Algorithm 1, and hence we can conclude that `NextClosures` is sound and complete. The following corollary summarises our results obtained so far, and also shows termination.

Corollary 6. *If the algorithm* `NextClosures` *is started on a finite formal context* $\mathbb{K}$ *as input, then it terminates, and returns both the set of all formal concepts and the canonical base of* $\mathbb{K}$ *as output.*

Proof. The second part of the statement is a direct consequence of Proposition 5. In the final state $|M|$ the set $\mathcal{L}$ contains all formal implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$. In particular, $\mathcal{L}$ is the canonical implicational base. Furthermore, the set $\mathfrak{B}$ contains all formal concepts of $\mathbb{K}$.

Finally, the computation time between states k and $k+1$ is finite, because there are only finitely many candidates of cardinality $k+1$, and the computation of closures w.r.t. the operators $\mathcal{L}^*$ and $\cdot^{II}$ can be done in finite time. As there are exactly $|M|$ states for a finite formal context, the algorithm must terminate. □

One could ask whether there are formal contexts that do not allow for a speedup in the enumeration of all intents and pseudo-intents on parallel execution. This would happen for formal contexts whose intents and pseudo-intents are linearly ordered. However, this is impossible.

Lemma 7. *Let $\mathbb{K} = (G, M, I)$ be a non-empty clarified formal context. Then the set of its intents and pseudo-intents is not linearly ordered w.r.t. subset inclusion $\subseteq$.*

Proof. Assume that $\mathbb{K} \coloneqq (G, M, I)$ with $G \coloneqq \{ g_1, \ldots, g_n \}$, $n > 0$, were a clarified formal context with intents and pseudo-intents $P_1 \subsetneq P_2 \subsetneq \ldots \subsetneq P_\ell$. In particular, then also all object intents form a chain $g_1^I \subsetneq g_2^I \subsetneq \ldots \subsetneq g_n^I$ where $n \leq \ell$. Since $\mathbb{K}$ is attribute-clarified, it follows $\left| g_{j+1}^I \setminus g_j^I \right| = 1$ for all j, and hence w.l.o.g. $M = \{ m_1, \ldots, m_n \}$, and $g_i \; I \; m_j$ iff $i \geq j$. Eventually, $\mathbb{K}$ is isomorphic to the ordinal scale $\mathbb{K}_n \coloneqq (\{ 1, \ldots, n \}, \{ 1, \ldots, n \}, \leq)$. It is easily verified that the pseudo-intents of $\mathbb{K}_n$ are either $\emptyset$, or of the form $\{ m, n \}$ where $m < n - 1$, a contradiction.

Consequently, there is no formal context with a linearly ordered set of intents and pseudo-intents. Hence, a parallel enumeration of the intents and pseudo-intents will always result in a speedup compared to a sequential enumeration.

4 Benchmarks

The purpose of this section is to show that our parallel algorithm for computing the canonical base indeed yields a speedup, both qualitatively and quantitatively, compared to the classical algorithm based on `NextClosure` [6]. To this end, we shall present the running times of our algorithm when applied to selected data-sets and with a varying number of available CPUs. We shall see that, up to a certain limit, the running time of our algorithms decreases proportional to the number of available CPUs. Furthermore, we shall also show that this speedup is not only qualitative, but indeed yields a real speedup compared to the original sequential algorithm for computing the canonical base.

The presented algorithm `NextClosures` has been integrated into *Concept Explorer FX* [8]. The implementation is a straight-forward adaption of Algorithm 1 to the programming language Java 8, and heavily uses the new Stream API and thread-safe concurrent collection classes (like `ConcurrentHashMap`). As we have described before, the processing of all candidates on the current cardinality level can be done in parallel, i.e. for each of them a separate thread is started that executes the necessary operations for lines 4 to 11 in Algorithm 1. Furthermore, as the candidates on the same level cannot affect each other, no communication between the threads is needed. More specifically, we have seen that the decision whether a candidate is an intent or a pseudo-intent is independent of all other sets with the same or a higher cardinality.

The formal contexts used for the benchmarks [1] are listed in Figure 1, and are either obtained from the *FCA Data Repository* [4] (ⓐ to ⓓ, and ⓕ to ⓟ), randomly created

[1] Readers who are interested in the test contexts should send a mail to one of the authors.

Formal Context	Objects	Attributes	Density
ⓐ car.cxt	1728	25	28 %
ⓑ mushroom.cxt	8124	119	19 %
ⓒ tic-tac-toe.cxt	958	29	34 %
ⓓ wine.cxt	178	68	20 %
ⓔ algorithms.cxt	2688	54	22 %
ⓕ o1000a10d10.cxt	1000	10	10 %
ⓖ o1000a20d10.cxt	1000	20	10 %
ⓗ o1000a36d17.cxt	1000	36	16 %
ⓘ o1000a49d14.cxt	1000	49	14 %
ⓙ o1000a50d10.cxt	1000	50	10 %
ⓚ o1000a64d12.cxt	1000	64	12 %
ⓛ o1000a81d11.cxt	1000	81	11 %
ⓜ o1000a100d10-001.cxt	1000	100	11 %
ⓝ o1000a100d10-002.cxt	1000	100	11 %
ⓞ o1000a100d10.cxt	1000	100	11 %
ⓟ o2000a81d11.cxt	2000	81	11 %
ⓠ 24.cxt	17	26	51 %
ⓡ 35.cxt	18	24	43 %
ⓢ 51.cxt	26	17	76 %
ⓣ 54.cxt	20	20	48 %
ⓤ 79.cxt	25	26	68 %

Fig. 1. Formal Contexts in Benchmarks

(ⓠ to ⓝ), or created from experimental results (ⓔ). For each of them we executed the implementation at least three times, and recorded the average computation times. The experiments were performed on the following two systems:

Taurus (1 Node of Bull HPC-Cluster, ZIH)
CPUs: 2x Intel Xeon E5-2690 with eight cores @ 2.9 GHz, RAM: 32 GB
Atlas (1 Node of Megware PC-Farm, ZIH)
CPUs: 4x AMD Opteron 6274 with sixteen cores @ 2.2 GHz, RAM: 64 GB

The benchmark results are displayed in Figure 2. The charts have both axes logarithmically scaled, to emphasise the correlation between the execution times and the number of available CPUs. We can see that the computation time is almost inverse linear proportional to the number of available CPUs, provided that the context is large enough. In this case there are enough candidates on each cardinality level for the computation to be done in parallel. However, we shall note that there are some cases where the computation times increase when utilising all available CPUs. We are currently not aware of an explanation for this exception – maybe it is due to some technical details of the platforms or the operation systems, e.g. some background tasks that are executed during the benchmark, or overhead caused by thread maintenance. Note that we did not have full system access during the experiments, but could only execute tasks by scheduling them in a batch system. Additionally, for some of the test contexts only benchmarks for a large number of CPUs could be performed, due to the time limitations on the test systems.

Furthermore, we have performed the same benchmark with small-sized contexts having at most 15 attributes. The computation times were far below one second. We have noticed that there is a certain number of available CPUs for which there is no

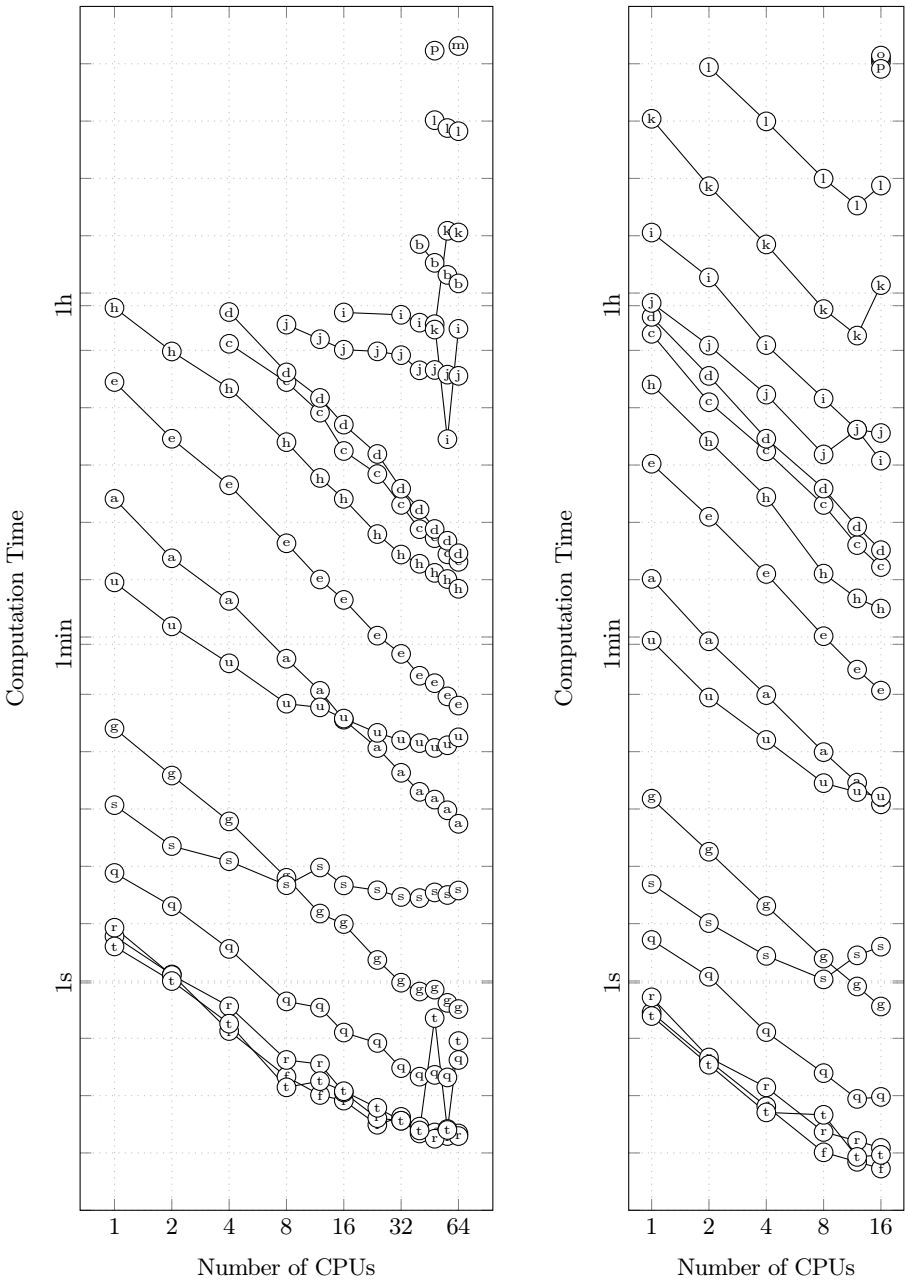

Fig. 2. Benchmark Results (left: Atlas, right: Taurus)

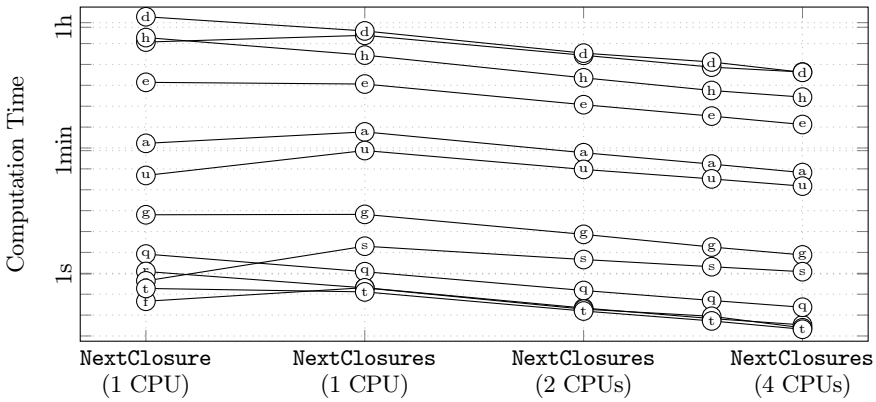

Fig. 3. Performance Comparison

further increase in speed of the algorithm. This happens when the number of candidates is smaller than the available CPUs.

Finally, we compared our two implementations of `NextClosure` and `NextClosures` when only one CPU is utilised. The comparison was performed on a notebook with Intel Core i7-3720QM CPU with four cores @ 2.6 GHz and 8 GB RAM. The results are shown in Figure 3. We conclude that our proposed algorithm is on average as fast as `NextClosure` on the test contexts. The computation time ratio is between $\frac{1}{3}$ and 3, depending on the specific context. Low or no speedups are expected for formal contexts where `NextClosure` does not have to do backtracking, and hence can find the next intent or pseudo-intent immediately.

5 Conclusion

In this paper we have introduced the parallel algorithm `NextClosures` for the computation of the canonical base. It constructs the lattice of all intents and pseudo-intents of a given formal context from bottom to top in a level-wise order w.r.t. increasing cardinality. As the elements in a certain level of this lattice can be computed independently, they can also be enumerated in parallel, thus yielding a parallel algorithm for computing the canonical base. Indeed, first benchmarks show that `NextClosures` allows for a speedup that is proportional to the number of available CPUs, up to a certain natural limit. Furthermore, we have compared its performance to the well-known algorithm `NextClosure` when utilising only one CPU. It could be observed that on average our algorithm (on one CPU) has the same performance as `NextClosure`, at least for the test contexts.

So far we have only introduced the core idea of the algorithm, but it should be clear that certain extensions are possible. For example, it is not hard to see that our parallel algorithm can be extended to also handle *background knowledge* given as a set of implications or as a *constraint closure operator* [1]. In order to yield *attribute exploration*, our algorithm can also be extended to include *expert interaction* for exploration of the canonical base of partially known contexts, much in the same way as the classical algorithm. One benefit is the possibility to have several experts answering questions in parallel. Another advantage is the constant increase in the difficulty of the questions (i.e. premise cardi-

nality), compared to the questions posed by default attribute exploration in lectic order. Those extensions have not been presented here due to a lack of space, but we shall present them in a future publication. Meanwhile, they can be found in a technical report [9].

Acknowledgements The authors thank Bernhard Ganter for helpful hints on optimal formal contexts for his `NextClosure` algorithm. Furthermore, the authors thank the anonymous reviewers for their constructive comments.

The benchmarks were performed on servers at the Institute of Theoretical Computer Science, and the Centre for Information Services and High Performance Computing (ZIH) at TU Dresden. We thank them for their generous allocations of computer time.

References

[1] Radim Belohlávek and Vilém Vychodil. "Formal Concept Analysis with Constraints by Closure Operators". In: *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures, ICCS 2006, Aalborg, Denmark, July 16-21, 2006, Proceedings*. Ed. by Henrik Schärfe, Pascal Hitzler, and Peter Øhrstrøm. Vol. 4068. Lecture Notes in Computer Science. Springer, 2006, pp. 131–143.

[2] Felix Distel. "Hardness of Enumerating Pseudo-Intents in the Lectic Order". In: *Proceedings of the 8th Interational Conference of Formal Concept Analysis*. (Agadir, Morocco). Ed. by Léonard Kwuida and Barış Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 124–137.

[3] Felix Distel and Barış Sertkaya. "On the Complexity of Enumerating Pseudo-Intents". In: *Discrete Applied Mathematics* 159.6 (2011), pp. 450–466.

[4] *FCA Data Repository*. URL: `http://www.fcarepository.com`.

[5] Huaiguo Fu and Engelbert Mephu Nguifo. "A Parallel Algorithm to Generate Formal Concepts for Large Data". In: *Proceedings of the Second International Conference on Formal Concept Analysis*. (Sydney, Australia). Ed. by Peter W. Eklund. Vol. 2961. Lecture Notes in Computer Science. Springer, 2004, pp. 394–401.

[6] Bernhard Ganter. "Two Basic Algorithms in Concept Analysis". In: *Proceedings of the 8th Interational Conference of Formal Concept Analysis*. (Agadir, Morocco). Ed. by Léonard Kwuida and Barış Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 312–340.

[7] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[8] Francesco Kriegel. *Concept Explorer FX*. Software for Formal Concept Analysis. 2010-2015. URL: `https://github.com/francesco-kriegel/conexp-fx`.

[9] Francesco Kriegel. *NextClosures – Parallel Exploration of Constrained Closure Operators*. LTCS-Report 15-01. Chair for Automata Theory, TU Dresden, 2015.

[10] Sergei O. Kuznetsov. "On the Intractability of Computing the Duquenne-Guigues Base". In: *Journal of Universal Computer Science* 10.8 (2004), pp. 927–933.

[11] Christian Lindig. "Fast Concept Analysis". In: *Working with Conceptual Structures – Contributions to ICCS 2000*. (Aachen, Germany). Ed. by Gerhard Stumme. Shaker Verlag, 2000, pp. 152–161.

[12] Sergei A. Obiedkov and Vincent Duquenne. "Attribute-Incremental Construction of the Canonical Implication Basis". In: *Annals of Mathematics and Artificial Intelligence* 49.1-4 (2007), pp. 77–99.

[13] Vilém Vychodil, Petr Krajča, and Jan Outrata. "Parallel Recursive Algorithm for FCA". In: *Proceedings of the 6th International Conference on Concept Lattices and Their Applications*. Ed. by Radim Bělohlávek and Sergej O. Kuznetsov. Palacký University, Olomouc, 2008, pp. 71–82.