Chapter 5

Implications

Have another look at the concept lattice shown in Figure 2.1 (p. 21). The six attributes describe how two unit squares can be placed with respect to each other. Each of the ten objects is a pair of unit squares, representing a possible placement. These ten pairs are representatives for an infinite set of possible positions that such pairs of squares may have. It is not stated, but perhaps expected by the reader, that these ten examples cover all possible combinations of the given attributes.

Such a situation occurs often: attributes are given, but objects are not known, or too many to handle them completely. We then have to study the possible attribute combinations, the *attribute logic* of the respective situation.

Let M be some set. We shall call the elements of M *attributes*, so as if we consider a formal context (G, M, I). However we do not assume that such a context is given or explicitly known.

Definition 28 An **implication between attributes** in M is a pair of subsets of M, denoted by $A \to B$. The set A is the **premise** of the implication $A \to B$, and B is its **conclusion**.

A subset $T \subseteq M$ **respects** an implication $A \to B$ if $A \not\subseteq T$ or $B \subseteq T$. We then also say that T is a **model** of the implication $A \to B$, and denote this by $T \models A \to B$. T **respects a set** $\mathcal{L}$ of implications if T respects every single implication in $\mathcal{L}$. The implication $A \to B$ **holds** in a set $\{T_1, T_2, \ldots\}$ of subsets if each of these subsets respects $A \to B$. With

$$\mathrm{Imp}\{T_1, T_2, \ldots\}$$

we denote the set of all implications that hold in $\{T_1, T_2, \ldots\}$. $\Diamond$

5.1 Implications of a formal context

Now let us consider the special case of implications of a formal context.

Definition 29 $A \to B$ **holds in a context** (G, M, I) if every object intent respects $A \to B$, that is, if each object that has all the attributes in A also has all the attributes in B. We then also say that $A \to B$ *is an implication of* (G, M, I). $\Diamond$

Proposition 25 *An implication $A \to B$ holds in (G, M, I) if and only if $B \subseteq A''$, which is equivalent to $A' \subseteq B'$. It then automatically holds in the set of all concept intents as well.*

An implication $A \to B$ holds in (G, M, I) if and only if each of the implications

$$A \to m, \qquad m \in B,$$

holds ($A \to m$ is short for $A \to \{m\}$). We can read this off from a concept lattice diagram in the following manner: $A \to m$ holds if the infimum of the attribute concepts corresponding to the attributes in A is less or equal than the attribute concept for m, formally if

$$\bigwedge\{\mu a \mid a \in A\} \le \mu m.$$

$A \to B$ holds in (G, M, I) if

$$\bigwedge\{\mu a \mid a \in A\} \le \bigwedge\{\mu b \mid b \in B\}.$$

5.2 Semantic and syntactical implication inference

As we will see, it is not necessary to store all implications of a formal context. We will discuss how implications can be derived from already known implications. First we discuss which kind of inference we want to model. This is fiven by the so-called *semantical inference*. Then we discuss a calculus (*syntactic inference*, and argue that the calculus is correct and complete with respect to our semantics.

5.2.1 When does an implication follow from other implications (semantically)?

Proposition 26 *If $\mathcal{L}$ is a set of implications in M, then*

$$\operatorname{Mod}\mathcal{L} := \{T \subseteq M \mid T \textit{ respects } \mathcal{L}\}$$

is a closure system on M. If $\mathcal{L}$ is the set of all implications of a context, then $\operatorname{Mod}\mathcal{L}$ *is the system of all concept intents.*

The respective closure operator

$$X \mapsto \mathcal{L}(X)$$

can be described as follows: For a set $X \subseteq M$, let

$$X^{\mathcal{L}} := X \cup \bigcup\{B \mid A \to B \in \mathcal{L}, A \subseteq X\}.$$

Form the sets $X^{\mathcal{L}}$, $X^{\mathcal{L}\mathcal{L}}$, $X^{\mathcal{L}\mathcal{L}\mathcal{L}}$, ... until[1] a set $\mathcal{L}(X) := X^{\mathcal{L}\cdots\mathcal{L}}$ is obtained with $\mathcal{L}(X)^{\mathcal{L}} = \mathcal{L}(X)$. Later on we shall discuss how to do this computation efficiently.

It is not difficult to construct, for any given set $\mathcal{L}$ of implications in M, a formal context such that $\operatorname{Mod}\mathcal{L}$ is the set of concept intents of this formal context. In fact, $(\operatorname{Mod}\mathcal{L}, M, \ni)$ will do.

Definition 30 An implication $A \to B$ **follows (semantically)** from a set $\mathcal{L}$ of implications in M if each subset of M respecting $\mathcal{L}$ also respects $A \to B$. A family of implications is called **closed** if every implication following from $\mathcal{L}$ is already contained in $\mathcal{L}$. A set $\mathcal{L}$ of implications of (G, M, I) is called **complete**, if every implication that holds in (G, M, I) follows from $\mathcal{L}$. $\Diamond$

In other words: An implication $A \to B$ follows semantically from $\mathcal{L}$ if it holds in every model of $\mathcal{L}$.

[1] If M is infinite, this may require infinitely many iterations.

Closure systems as extents

As an exercise in abstraction we demonstrate that the situation we have just discussed can neatly be formulated in formal context language. For a given set M, we may define a formal context

$$(\mathfrak{P}(M), \{A \to B \mid A, B \subseteq M\}, \models).$$

The attributes of this formal context are all implications in M, the objects are all subsets of M, and $T \models A \to B$ says that the subset T is a models of the implication $A \to B$. The derivation operators of this context are the operators Imp and Mod. The concept intents are precisesly the complete sets of implications in M, the corresponding extents are precisely the closure systems on M. Therefore, the concept lattice of this formal context is isomorphic to the lattice of all closure systems on M, and dually isomorphic to the lattice of all closed implication sets on M.

We give an example for $M := \{a, b, c\}$, but display, for simplicity, only the *reduced* context. We omit some of the set brackets.

	$\emptyset \to a$	$\emptyset \to b$	$\emptyset \to c$	$a \to b$	$a \to c$	$b \to a$	$b \to c$	$c \to a$	$c \to b$	$a,b \to c$	$a,c \to b$	$b,c \to a$
$\emptyset$	.	.	.	×	×	×	×	×	×	×	×	×
$\{a\}$	×	.	.	.	.	×	×	×	×	×	×	×
$\{b\}$	.	×	.	×	×	.	.	×	×	×	×	×
$\{a, b\}$	×	×	.	×	.	×	.	×	×	.	×	×
$\{c\}$	.	.	×	×	×	×	×	.	.	×	×	×
$\{a, c\}$	×	.	×	.	×	×	×	×	.	×	.	×
$\{b, c\}$	.	×	×	.	×	.	×	.	×	×	×	.

Note that the extents of this formal context form a closure system, while each extent itself is a closure system. What we get is the *closure system of all closure systems on M*. Even more brain-twisting seems the question which implications hold in this context. The attributes themselves are implications. So we get implications between sets of implications. We shall come back to that later.

5.2.2 When does an implication follow from other implications (syntactically)?

The semantic definition of implication inference has a syntactic counterpart. We can give sound and complete inference rules (known as **Armstrong rules** [?]) and an efficient algorithm for inference testing.

Proposition 27 *A set $\mathcal{L}$ of implications in M is closed if and only if the following conditions are satisfied for all $W, X, Y, Z \subseteq M$:*

1. *$X \to X \in \mathcal{L}$,*
2. *If $X \to Y \in \mathcal{L}$, then $X \cup Z \to Y \in \mathcal{L}$,*
3. *If $X \to Y \in \mathcal{L}$ and $Y \cup Z \to W \in \mathcal{L}$, then $X \cup Z \to W \in \mathcal{L}$.*

Readers with a background in Computational Logic may prefer a different notation of these Armstrong rules:

$$\frac{}{X \to X}, \qquad \frac{X \to Y}{X \cup Z \to Y}, \qquad \frac{X \to Y, \quad Y \cup Z \to W}{X \cup Z \to W}.$$

The proposition says that a set of implications is the set of all implications of some context if and only if it is closed with respect to these rules. In other words, an implication follows from other implications if and only if it can be derived from these by successive applications of these rules. In particular, semantic and syntactic inference are the same.

However, these rules do not always suggest the best proof strategy. Instead, we may note the following:

Proposition 28 *An implication $X \to Y$ follows from a list $\mathcal{L}$ of implications if and only if $Y \subseteq \mathcal{L}(X)$.*

Proof $\mathcal{L}(X)$ is a model of $\mathcal{L}$ containing X. If $Y \not\subseteq \mathcal{L}(X)$, then $\mathcal{L}(X)$ is no model of $X \to Y$. Then, $X \to Y$ does not follow from $\mathcal{L}$. Conversely, any model of $\mathcal{L}$ that contains X must also contain $X^{\mathcal{L}}$ and, by induction, also contain $\mathcal{L}(X)$, which proves the Proposition. □

The Closure Algorithm

We give an algorithm that efficiently computes the closure $\mathcal{L}(X)$ for any given set X. Such algorithms are used in the theory of relational data bases for the study of functional dependencies. Later on, we shall show that this can be understood as a special instance of our approach.

```
Algorithm CLOSURE
Input:  A list $\mathcal{L} =: [\mathcal{L}[1], \ldots, \mathcal{L}[n]]$ of implications in $M$
         and a set $X \subseteq M$.
Output:  The closure $\mathcal{L}(X)$ of $X$.
begin
  for all $x \in M$ do
  begin
    avoid[$x$] := $\{1, \ldots, n\}$;
    for $i := 1$ to $n$ do with $A \to B := \mathcal{L}[i]$
      if $x \in A$ then avoid[$x$] :=avoid[$x$] $\setminus \{i\}$;
  end;
  used_imps :=$\emptyset$;
  old_closure :=$\{-1\}$;     (* some element not in $M$ *);
  new_closure := $X$;
  while new_closure $\neq$ old_closure do
  begin
    old_closure := new_closure;
    $T := M \setminus$new_closure;
    usable_imps := $\bigcap_{x \in T}$avoid[$x$];
    use_now_imps := usable_imps $\setminus$ used_imps;
    used_imps := usable_imps;
    for all $i \in$ use_now_imps with $A \to B := \mathcal{L}[i]$ do
      new_closure := new_closure $\cup B$;
  end;
  $\mathcal{L}(x)$ :=new_closure;
end.
```

Figure 5.1: Algorithm CLOSURE.

We can give a rough complexity estimation of the algorithm in Figure 5.1. Except for manipulations of addresses, the main effort is to apply the implications. Each implication is applied at most once, and each application requires a simple set operation. Therefore the time required by the Linclosure algorithm is essentially *linear* in the size of the input $\mathcal{L}$.

Linear complexity of implication inference

Summarizing these considerations we learn that implication inference is easy: to check if an implication $X \to Y$ follows from a list $\mathcal{L}$ of implications, it suffices to check if $Y \subseteq \mathcal{L}(X)$ (by Proposition 28), and this can be done in time linear in the size of the input.

In other words: implications are easy to use, much easier than many other logical constructs. This may be a reason why implications are popular, and perhaps be part of an explanation why our simple theory of formal concepts is so useful.

5.3 The Stem Base

The number of implications that hold in a given situation can be very large. For example, if there is only one closed set, M, then *every* implication holds. If M has n elements, then these are some 2^{2n} implications. But this is ridiculous, because all these implications can be inferred from a single one, namely from $\emptyset \to M$.

We see from this trivial example that the set of *all* implications of a given formal context may be highly redundant. It is a natural question to ask for a small *implicational base*, from which everything else follows. More precisely we ask, for any given formal context (G, M, I), for a list $\mathcal{L}$ of implications that is

- sound (i.e., each implication in $\mathcal{L}$ holds in (G, M, I)),
- complete (i.e., each implication that holds in (G, M, I) follows from $\mathcal{L}$), and
- non redundant (i.e., no implication in $\mathcal{L}$ follows from other implications in $\mathcal{L}$).

It is easy to see that (for finite M) such sets $\mathcal{L}$ exist. We may start with some sound and complete set of implications, for example, with the set of all implications that hold in (G, M, I). We then can successively remove redundant implications from this set, until we obtain a sound, complete, non redundant set.

But this is and unrealistic procedure. We therefore look for a better way to construct an implicational base. Duquenne and Guigues [?] have shown that there is a natural choice, the *stem base*.

5.3.1 A recursive definition

The following recursive definition is rather irritating on the first glance. We define a pseudo closet set to be a set which is not closed, but contains the closure of every pseudo-closed proper subset. More precisely,

Definition 31 Let $X \mapsto X''$ be a closure operator on the finite set M. We call a subset $P \subseteq M$ **pseudo-closed**, if (and only if)

- $P \neq P''$, and
- if $Q \subset P$ is a pseudo-closed proper subset of P, then $Q'' \subseteq P$.

$\Diamond$

This *is* a valid recursive definition. It is not circular, because the pseudo-closed set Q must have fewer elements than P, because it is a proper subset.[2]

Reformulating this definition for formal contexts, we obtain

Definition 32 Let (G, M, I) be a formal context with M finite. A subset $P \subseteq M$ is a **pseudo intent** of (G, M, I) iff

- P is not a concept intent, and
- if $Q \subset P$ is a pseudo-closed proper subset of P, then there is some object $g \in G$ such that $Q \subseteq g'$ but $P \not\subseteq g'$.

$\Diamond$

5.3.2 The stem base

Theorem 29 *Let M be finite and let $X \mapsto X''$ be some closure operator on M. The set of implications*

$$\{P \to P'' \mid P \textit{ pseudo-closed}\}$$

is sound, complete, and non redundant.

The **Proof** is so simple that we can give it here. Obviously all implications of the form $X \to X''$ hold in the given closure system, therefore the set is sound. Suppose it were not complete. Then there is some implication $X \to Y$ that holds, but does not follow from the set $\mathcal{L}$ of implications given in the theorem. In particular, $Y \subseteq X''$ (because the implication holds) but $Y \not\subseteq \mathcal{L}(X)$ (because the implication does not follow). Then $\mathcal{L}(X) \neq X''$, and thus $\mathcal{L}(X)$ is not closed. Now let $Q \subset \mathcal{L}(X)$ be any pseudo-closed proper subset of $\mathcal{L}(X)$. Since the implication $Q \to Q''$ belongs to $\mathcal{L}$, Q'' must be in $\mathcal{L}(X)$.

Therefore, $\mathcal{L}(X)$ is not closed but contains the closure of every pseudo closed proper subset. By definition then $\mathcal{L}(X)$ must be pseudo-closed. But that gives a contradiction, because then the implication $\mathcal{L}(X) \to \mathcal{L}(X)''$ is in $\mathcal{L}$, which implies $\mathcal{L}(X)'' \subseteq \mathcal{L}(X)$ and consequently $\mathcal{L}(X) = \mathcal{L}(X)''$.

To see that the list $\mathcal{L}$ is non redundant, remove one of the implications, say, $P \to P''$. With respect to the smaller list $\mathcal{L}^-$ then P is closed (because P respects all implications $Q \to Q''$ in $\mathcal{L}^-$), and thus $\mathcal{L}^-(P) = P$. By Proposition 28, $P \to P''$ does not follow from $\mathcal{L}^-$. $\square$

The implication set in the theorem deserves a name. It is sometimes called the *Duquenne–Guigues–base.* We simply call it the **stem base** of the closure operator (or the *stem base of a given formal context*, if the closure operator is given that way). In practice one uses a slightly different version of the stem base, namely

$$\{P \to P'' \setminus P \mid P \text{ pseudo-closed}\}.$$

The stem base is not the only implicational base, but it plays a special rôle. For example, no implicational base can consist of fewer implications, as the next proposition shows:

Proposition 30 *Every sound and complete set of implications contains, for every pseudo closed set P, an implication $A \to B$ with $A'' = P''$.*

[2]'Recursive' is meant here with respect to set inclusion. Compare with the following recursive definition: A natural number is *prime* iff it is greater than 1 and not divisible by any smaller prime number.

5.3.3 ($*$) Some open questions about pseudo-closed sets.

Pseudo closed sets are not well understood. It is not even known how many there are. More precisely: It is easy to give an example of a closure system with many pseudo-closed sets. Take, for a given base set M with $|M| = 2n > 2$, as closed sets all subsets with at most n elements, plus M itself. The pseudo-closed sets then are precisely the subsets with $n + 1$ elements. There are $\binom{2n}{n}$ of them, a number that is exponential in n. Thus the number of pseudo-closed sets can be exponential in comparison with the size of the base set.

But any formal context describing this closure system must have at least $\binom{2n}{n}$ objects, and thus itself be of a size exponential in n. In other words, the number of pseudo-closed sets in this example is large in comparison with the base set, but small in comparison with the formal context. We do not know if there are (infinite series of) contexts with *many* pseudo-closed sets, meaning that the number of pseudo-closed sets is exponential in the size of the formal contexts.

Another unresolved problem is how difficult it is to decide if a given set is pseudo-closed.

Is the following problem in $\mathcal{NP}$?

INSTANCE: A formal context (G, M, I) and a set $P \subseteq M$.

TASK: Decide if P is pseudo-closed.

Figure 5.2: An open problem on pseudo-closed sets

5.3.4 Making stem base implications shorter

We have stated above that the stem base is non redundant. This means that no implication in the stem base is dispensable. It is however possible that an implication in the base may be shortened, for example because it has a redundant premise. Such are occasionally counter–intuitive, in particular for mathematicians, who are trained to base their proofs on minimal resources. Similarly, the conclusion of an implication may be redundant. It may suffice to prove only part of it, and then to use other implications to obtain the rest.

It seems to be straightforward what to do: if an attribute is redundant in a premise or a conclusion, just omit it. But things get more complicated if two or more attributes are redundant. If each of m and n is redundant, it does not mean that they may be omitted *simultaneously*. In general, there is no guarantee that a set A contains a *unique* minimal generating set $S \subseteq A$ satisfying $A'' = S''$. There may be many such sets, and they may be difficult to find. For example, to find such a set with minimum possible cardinality is an $\mathcal{NP}$–hard problem. This can be derived from the following observations:

(to be completed . . .)

5.4 Computing the Stem Base

As before, consider a closure operator $X \mapsto X''$ on a finite set M. We start with a harmless definition:

Definition 33 A set $Q \subseteq M$ is **•-closed** if it contains the closure of every •-closed set that is properly contained in Q.

Formally, Q is $\bullet$-closed iff for each $\bullet$-closed set $Q_0 \subset Q$ with $Q_0 \neq Q$ we have $Q_0'' \subseteq Q$. $\Diamond$

This is a simple (but convenient) renaming, as the next proposition shows.

Proposition 31 *A set is $\bullet$-closed iff it is either closed or pseudo-closed.*

Observe that if Q contains the closure of every $\bullet$-closed subset, then Q must be closed.

5.4.1 $\bullet$-closed sets form a closure system

The first crucial step towards finding pseudo-closed sets is this:

Proposition 32 *The intersection of $\bullet$-closed sets is $\bullet$-closed.*

Proof Let Q_t, $t \in T$, be $\bullet$-closed and let $Q := \bigcap_{t\in T} Q_t$. We have to show that Q is $\bullet$-closed. Actually, we show more: Either $Q = Q_t$ for some $t \in T$, or Q is closed. Assume $Q \neq Q_t$ for all t, and let Q_0 $(0 \notin T)$ be some $\bullet$-closed set contained in Q. Then Q_0 is properly contained in each Q_t, $t \in T$, because Q is. Since each Q_t is $\bullet$-closed, the closure of Q_0 must also be contained in Q. Therefore Q is closed. $\Box$

In other words: the $\bullet$-closed sets form a closure system. We have described an algorithm to compute, for a given closure operator, all closed sets. We can apply this algorithm for computing all $\bullet$-closed sets, provided that we can access the corresponding closure operator. This is easy. We prepare the result with a proposition that is an immediate consequence of Definition 33.

Proposition 33 *Q is $\bullet$-closed iff Q satisfies the following condition:*

If $P \subset Q$, $P \neq Q$, is pseudo-closed, then $P'' \subseteq Q$.

5.4.2 The closure operator

Proposition 33 shows how to find the quasi closure of an arbitrary set $S \subseteq M$: As long as the condition in the proposition is violated, we (are forced to) extend the set S, until we finally reach a fixed point.

Let $\mathcal{L}$ be the stem base[3]. Define, for $X \subseteq M$,

$$X^{\mathcal{L}^\bullet} := X \cup \bigcup\{P'' \mid P \to P'' \in \mathcal{L}, P \subset X, P \neq X\},$$

iterate by forming

$$X^{\mathcal{L}^\bullet\mathcal{L}^\bullet}, X^{\mathcal{L}^\bullet\mathcal{L}^\bullet\mathcal{L}^\bullet}, \ldots$$

until a set

$$\mathcal{L}^\bullet(X) := X^{\mathcal{L}^\bullet\mathcal{L}^\bullet\ldots\mathcal{L}^\bullet}$$

is obtained that satisfies

$$\mathcal{L}^\bullet(X) = \mathcal{L}^\bullet(X)^{\mathcal{L}^\bullet}.$$

Proposition 34 *$\mathcal{L}^\bullet(X)$ is the smallest $\bullet$-closed set containing X.*

Note that in order to find the quasi closure, we use only pseudo-closed sets which are contained in the closure and therefore in particular are lectically smaller than the quasi closure. Thus, the same result is obtained if $\mathcal{L}$ is a subset of the stem base, containing thoses implications $P \to P''$ for which P is pseudo-closed and lectically smaller than $\mathcal{L}^\bullet(X)$.

[3]The reader wonder why we use the stem base to construct the stem base. As we shall see soon, this works, due to the recursive definition of the stem base.

```
Algorithm $\mathcal{L}^\bullet$–Closure
Input:  A list $\mathcal{L} =: [\mathcal{L}[1], \dots, \mathcal{L}[n]]$ of implications in $M$
        and a set $X \subseteq M$.
Output: The closure $\mathcal{L}(X)$ of $X$.
begin
  for all $x \in M$ do
  begin
    avoid$[x] := \{1, \dots, n\}$;
    for $i := 1$ to $n$ do with $A \to B := \mathcal{L}[i]$
      if $x \in A$ then avoid$[x]$ :=avoid$[x] \setminus \{i\}$;
  end;
  used_imps :=$\emptyset$;
  old_closure :=$\{-1\}$;     (* some element not in $M$ *);
  new_closure := $X$;
  while new_closure $\neq$ old_closure do
  begin
    old_closure := new_closure;
    $T := M \setminus$ new_closure;
    usable_imps := $\bigcap_{x \in T}$avoid$[x]$ $\cap \bigcup_{x \in \text{new\_closure}}$avoid$[x]$;
    use_now_imps := usable_imps $\setminus$ used_imps;
    used_imps := usable_imps;
    for all $i \in$ use_now_imps with $A \to B := \mathcal{L}[i]$ do
      new_closure := new_closure $\cup B$;
  end;
  $\mathcal{L}(x)$ :=new_closure;
end.
```

Figure 5.3: Computing the $\mathcal{L}^\bullet$–Closure.

5.4.3 An algorithm for computing the stem base

Now it is easy to give an algorithm to compute all pseudo-closed sets for a given closure operator. We use the Next Closure algorithm applied to the closure system of •-closed sets. For short, we shall refer to this as the **next quasi closure** after a given set A, NEXT_$\mathcal{L}^\bullet$_CLOSURE(A). This produces all •-closed sets in lectic order. We record only those which are not closed. This yields a list of all pseudo-closed sets.

Since the •-closed sets are generated in lectic order, we have, at each step, the full information about the lectically smaller pseudo-closed sets. We have seen that this suffices to compute the "quasi closure" operator. The algorithm in Figure 5.4 uses a dynamic list $\mathcal{L}$. Whenever a pseudo-closed set P is found, the corresponding implication $P \to P''$ is included in the list. Since the pseudo-closed sets and found in lectic order, this makes sure that at any step we have sufficient information to compute the quasi closure.

5.4.4 An example

We compute the stem base for the context of triangles given on page 27. The steps are shown in figure 5.5. The first column contains all quasi closed sets, in lectic order. The pseudo-closed sets are precisely those which are not closed (see middle column). Each pseudo-closed set gives rise to an entry in the stem base (last column, short form). This stem base is given again, in slightly modified form, in Figure 6.1 (p. 101).

```
Algorithm STEM BASE
Input:   A closure operator $X \mapsto X''$ on a finite set $M$,
         for example given by a formal context $(G, M, I)$.
Output:  The stem base $\mathcal{L}$

begin
  $\mathcal{L} := \emptyset$;
  $A := \emptyset$;
  while $A \neq M$ do
  begin
    if $A \neq A''$ then $\mathcal{L} := \mathcal{L} \cup \{A \to A''\}$;
    $A :=$ NEXT_$\mathcal{L}^\bullet$_CLOSURE$(A)$;
  end;
end.
```

Figure 5.4: Computing the stem base for a given closure operator.

Since the closure operator is given in terms of a formal context, we may speak of **quasi intent**s and **pseudo intent**s instead of •-closed sets or pseudo-closed sets. We see that the algorithm generates all quasi intents to find the stem base. In other words, to compute all pseudo intents we also compute all intents, possibly exponentially many. This looks like a rather unefficient method. Unfortunately, we do not know of a better strategy. It is an open problem to find a better algorithm for the stem base. In practice, the algorithm is not fast, but nevertheless very useful.

•-closed set	closed ?	stem base implication
$\emptyset$	yes	
$\{e\}$	yes	
$\{d\}$	yes	
$\{d,e\}$	no	$\{d,e\} \to \{a,b,c\}$
$\{c\}$	yes	
$\{c,e\}$	no	$\{c,e\} \to \{a,b,d\}$
$\{c,d\}$	no	$\{c,d\} \to \{a,b,e\}$
$\{b\}$	yes	
$\{b,e\}$	yes	
$\{b,d\}$	yes	
$\{b,c\}$	yes	
$\{a\}$	no	$\{a\} \to \{b,c\}$
$\{a,b,c\}$	yes	
$\{a,b,c,d,e\}$	yes	

Figure 5.5: Steps in the stem base algorithm

5.5 Database Dependencies

How can we apply the theory of implications in the case of many-valued contexts? The attribute implications in the derived context offer one approach, however elementary. Basically, it describes implications between the individual attribute values, at least as long as we keep to plain scaling.

In colloquial language we use the term **dependency** of many-valued attributes as exemplified by the following sentence

"The price of a real-estate depends on situation and size".

This is meant to express a simultaneous dependency of attribute values, perhaps even a gradual one, in the sense of "the larger the more expensive".

There are different notions of dependency for many-valued attributes, which correspond to the different possibilities of scaling. For an integration into a general theoretic framework, please refer to the corresponding literature.

We now describe the case of *functional dependency*, the (stronger) one of *ordinal dependency* and will indicate generalizations. For reasons of simplicity, we will first concentrate on complete many-valued contexts.

5.5.1 Functional dependencies

Definition 34 If $X \subseteq M$ and $Y \subseteq M$ are sets of attributes of a complete many-valued context (G, M, W, I), then we say that Y is **functionally dependent** on X if the following holds for every pair of objects $g, h \in G$:

$$(\forall_{m \in X}\ m(g) = m(h)) \Rightarrow (\forall_{n \in Y}\ n(g) = n(h)).$$

$\Diamond$

That is to say, if two objects have the same values with respect to all attributes from X the same must be true for the attributes from Y. This notion of dependency is often used in the theory of relational databases. The term "functional" can be explained as follows: Y is functionally dependent on X if and only if there is a map $f : W^X \to W^Y$ with

$$f(m(g) \mid m \in X) = (n(g) \mid n \in Y) \qquad \text{for all } g \in G.$$

In the case of ordinal dependency, we consider an ordinal context, i.e., we have for each attribute $m \in M$ an order $\leq_m$ on the set $m(G)$ of the values of m. (We obtain the special case of functional dependency if we take the equality relation for each of those orders.)

5.5.2 Ordinal dependencies

Definition 35 Let (G, M, W, I) be a complete many-valued context and let $\leq_m$ be an order relation on the set $m(G)$ of the values of m for every attribute $m \in M$. If $X \subseteq M$ and $Y \subseteq M$ are sets of attributes, we call Y **ordinally dependent** on X if the following holds for each pair of objects $g, h \in G$:

$$(\forall_{m \in X}\ m(g) \leq_m m(h)) \Rightarrow (\forall_{n \in Y}\ n(g) \leq_n n(h)).$$

$\Diamond$

Irrespective of which orders $\leq_m$ we have chosen, ordinal dependency always implies functional dependency, since from $m(g) = m(h)$ it follows that $m(g) \leq_m m(h)$ as well as $m(h) \leq_m m(g)$, and vice versa. Thus, one would expect that ordinal dependency is a kind of "order-preserving functional dependency". Intuitively, this is quite correct, but it is difficult to formulate, since the condition of being order-preserving is only required for the tuples $(m(g) \mid m \in X)$ that appear in the context. Not every map of this kind can be extended to form an order-preserving map of W^X to W^Y.

The ordinal dependencies (and as a special case within them the functional dependencies) of many-valued contexts can be expressed elegantly by implications of appropriate one-valued contexts. By means of the rule

$$(g,h) I_{\mathbb{O}}\, m \quad :\Longleftrightarrow\; m(g) \leq_m m(h),$$

we define a one-valued context

$$\mathbb{K}_{\mathbb{O}} := (G \times G, M, I_{\mathbb{O}})$$

for a complete many-valued context (G, M, W, I) with orders $\leq_m$ on the values. For the functional dependencies the context can be simplified further: It is possible to take advantage of the symmetry of the equality relation and to define

$$\mathbb{K}_{\mathbb{N}} := (\mathfrak{P}_2(G), M, I_{\mathbb{N}})$$

by

$$\{g,h\} I_{\mathbb{N}}\, m \quad :\Longleftrightarrow\; m(g) = m(h).$$

Then,

$$\mathfrak{P}_2(G) := \{\{g,h\} \mid g,h \in G, g \neq h\}.$$

The contexts defined in this way exactly fit the above-mentioned definitions of the dependencies and it is easy to prove the following proposition:

Corollary 35 *In (G, M, W, I) the attribute set Y is functionally dependent on X if and only if the implication $X \to Y$ holds in the context $\mathbb{K}_{\mathbb{N}}$. In (G, M, W, I) the attribute set Y is ordinally dependent on X if and only if the implication $X \to Y$ holds in the context $\mathbb{K}_{\mathbb{O}}$.* □

Hereby we have traced back the theory of functional and ordinal dependencies completely to the theory of implications. In particular, the algorithm mentioned in the previous section can also be used for the creation of a basis for the functional or ordinal dependencies, respectively.

The translation works even if the many-valued context (G, M, W, I) is not complete. In this connection, first of all we observe that Y is ordinally dependent on X if and only if this is true for every single attribute in Y, i.e., if $\{n\}$ is ordinally dependent on X for every $n \in Y$. This means that it is sufficient to state in which cases a single attribute is dependent on an attribute set. ... ———(to be written)———

5.6 Association rules

— This section is still in a preliminary form —

One of the core tasks of *Knowledge Discovery in Databases* (*KDD*) is the mining of association rules (conditional implications). *Association rules* are statements of the type '67 % of the customers buying cereals and sugar also buy milk (where 7% of all customers buy all three items)'. The task of mining association rules is to determine all rules whose *confidences* (67 % in the example) and *supports* (7 % in the example) are above user-defined thresholds. Since the problem was stated [2], various approaches have been proposed for an increased efficiency of rule discovery in very large databases [3, 8, 12, 31, 32]. However, fully taking advantage of exhibited rules means providing capabilities to handle them. The problem is especially critical when collected data is highly correlated or dense, like in statistical databases [12]. For instance, when applied to a census dataset of 10,000 objects, each of which characterized by values of 73 attributes, experiments result in more then 2,000,000 rules with support and confidence greater than or equal 90%. Thus the question arises: How can long lists of association rules be reduced in size?

Formal Concept Analysis allows to significantly reduce the number of rules without losing any information. We extract only a subset of all association rules, called *basis*, from which all other rules can be derived.

We use results of Duquenne and Guigues ([13], cf. also [16]) and Luxenburger [23, 24]. The former have studied bases (i. e., minimal non-redundant sets of rules from which all other rules can be derived) for association rules with 100 % confidence, and the latter association rules with less than 100 % confidence, but neither of them considered the support of the rules. We adopt their results to association rules (where both the support and the confidence are considered) and provide algorithms for computing the new bases by using *iceberg concept lattices* [40]. We follow an approach in two steps. In the first step, we compute the iceberg concept lattice for the given parameters. It consists of all FCA concepts whose extents exceed the user-defined minimum support. In the second step, we derive the bases for the association rules. In this paper, we focus on the second step. For the first step, we refer to the Pascal [7] and Titanic [39] algorithms.

This two-step approach has two advantages compared to the classical two-step approach [3] (which computes all frequent itemsets as intermediate result, and not only those which are intents of frequent FCA concepts):

1. It allows to determine bases for non-redundant association rules and thus to prune redundancy.

2. It speeds up the computation, especially for strongly correlated data or when the minimum support is low.

5.6.1 Formal Concept Analysis and the Association Rule Framework

In this section, we use (in the current version of this script) special notations. Therefore, we recall the basic definitions. ———(to be written)———

Definition 36 A *formal context* is a triple $\mathbb{K} := (G, M, R)$ where G and M are sets and $R \subseteq G \times M$ is a binary relation. A *data mining context* (or *dataset*) is a formal context where G and M are finite sets. Its elements are called *objects* and *items*, respectively. $(o, i) \in R$ is read as "object o is related to item i".

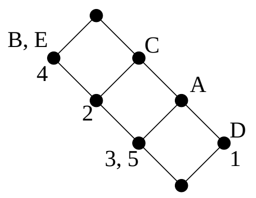
For $O \subseteq G$, we define $f(O) := \{i \in M \mid \forall o \in O\colon (o,i) \in R\}$; and for $I \subseteq M$, we define dually $g(I) := \{o \in G \mid \forall i \in I\colon (o,i) \in R\}$. A *formal concept* is a pair $(O, I) \in \mathfrak{P}(G) \times \mathfrak{P}(M)$ with $f(O) = I$ and $g(I) = O$. O is called *extent* and I is called *intent* of the concept. The set of all concepts of a formal context $\mathbb{K}$ together with the partial order $(O_1, I_1) \leq (O_2, I_2) :\Longleftrightarrow O_1 \subseteq O_2$ ($\Longleftrightarrow I_2 \subseteq I_1$) is a complete lattice, called *concept lattice* of $\mathbb{K}$.

In this setting, we call each subset of M also *itemset*, and each intent I also *closed itemset* (since it satisfies the equation $I = f(g(I))$). For two closed itemsets I_1 and I_2, we note $I_1 \prec I_2$ if $I_1 \subset I_2$ and if there does not exist a closed itemset I_3 with $I_1 \subset I_3 \subset I_2$.[4] $\Diamond$

In the following, we will use the composed function $h := f \circ g\colon \mathfrak{P}(M) \to \mathfrak{P}(M)$ which is a closure operator on M (i. e., it is extensive, monotonous, and idempotent). The related closure system (i. e., the set of all $I \subseteq M$ with $h(I) = I$) is exactly the set of the intents of all concepts of the context.

[4]We write $X \subset Y$ if and only if $X \subseteq Y$ and $X \neq Y$.

	A	B	C	D	E
1	×		×	×	
2		×	×		×
3	×	×	×		×
4		×			×
5	×	×	×		×



Exact rule	Supp	Approximate rule	Supp	Conf	Approximate rule	Supp	Conf	Approximate rule	Supp	Conf
ABC → E	0.4	BCE → A	0.4	2/3	C → ABE	0.4	2/4	C → BE	0.4	3/4
ABE → C	0.4	AC → BE	0.4	2/3	E → ABC	0.4	2/4	E → BC	0.6	3/4
ACE → B	0.4	BE → AC	0.4	2/4	A → BC	0.4	2/3	A → B	0.4	2/3
AB → CE	0.4	CE → AB	0.4	2/3	B → AC	0.4	2/4	B → A	0.4	2/4
AE → BC	0.4	AC → B	0.4	2/3	C → AB	0.4	2/4	C → A	0.6	3/4
AB → C	0.4	BC → A	0.4	2/3	A → BE	0.4	2/3	A → E	0.4	2/3
AB → E	0.4	BE → A	0.4	2/4	B → AE	0.4	2/4	E → A	0.4	2/4
AE → B	0.4	AC → E	0.4	2/3	E → AB	0.4	2/4	C → B	0.6	3/4
AE → C	0.4	CE → A	0.4	2/3	A → CE	0.4	2/3	C → E	0.6	3/4
BC → E	0.6	BE → C	0.6	3/4	C → AE	0.4	2/4	E → C	0.6	3/4
CE → B	0.6	A → BCE	0.4	2/3	E → AC	0.4	2/4	∅ → E	0.8	4/5
A → C	0.6	B → ACE	0.4	2/4	B → CE	0.6	3/4	∅ → BE	0.8	4/5
B → E	0.8	∅ → C	0.8	4/5	∅ → AC	0.6	3/5	∅ → BC	0.6	3/5
E → B	0.8	∅ → A	0.6	3/5	∅ → B	0.8	4/5	∅ → CE	0.6	3/5
								∅ → BCE	0.6	3/5

Figure 5.6: The example data mining context $\mathbb{K}$ and its concept lattice. The table shows all association rules that hold in $\mathbb{K}$ for *minsupp* $= 0.4$ and *minconf* $= 1/2$.

Definition 37 Let $I \subseteq M$, and let *minsupp*, *minconf* $\in [0,1]$. The *support count* of the itemset I in $\mathbb{K}$ is $supp(I) := \frac{|g(I)|}{|G|}$. I is said to be *frequent* if $\text{supp}(I) \geq$*minsupp*. The set of all frequent itemsets of a context is denoted FI.

An *association rule* is a pair of itemsets I_1 and I_2, denoted $I_1 \to I_2$, where $I_2 \neq \emptyset$. I_1 and I_2 are called *antecedent* and *consequent* of the rule, respectively. The *support* and *confidence* of an association rule $r := I_1 \to I_2$ are defined as follows: $\text{supp}(r) := \frac{|g(I_1 \cup I_2)|}{|G|}$, $\text{conf}(r) := \frac{supp(I_1 \cup I_2)}{supp(I_1)}$. If *conf(r)*=1, then r is called *exact association rule* (or *implication*), otherwise r is called *approximate association rule.*

An association rule r *holds* in the context if $\text{supp}(r) \geq$ minsupp and $\text{conf}(r) \geq$ minconf. The set of all association rules holding in $\mathbb{K}$ for given *minsupp* and *minconf* is denoted AR. $\Diamond$

Remark. The definition of association rules often includes the additional condition $I_1 \cap I_2 = \emptyset$. This condition helps pruning rules which are obviously redundant, as $I_1 \to I_2$ and $I_1 \to I_2 \setminus I_1$ have same support and same confidence. In this paper, we omit the condition, in order to simplify definitions. When discussing the algorithms, however, we will use the condition since it saves memory.

The association rule framework has first been formulated in terms of Formal Concept Analysis independently in [29], [38], and [43]. [29] provided also the first algorithm (named Close) based on this approach.

Example 7 An example data mining context $\mathbb{K}$ consisting of five objects (identified by their OID) and five items is given in Figure 5.6 together with its concept lattice. The association rules holding for *minsupp* $= 0.4$ and *minconf* $= 1/2$ are shown in the lower table.

In the *line diagram*, the name of an object g is always attached to the node representing the smallest concept with g in its extent; dually, the name of an attribute m is always attached to the node representing the largest concept with m in its intent. This allows us to read the context relation from the diagram because an object g has an attribute m if and only if there is an ascending path from the node labeled by g to the node labeled by m. The extent of a concept consists of all objects whose labels are below in the diagram, and the intent consists of all attributes attached to concepts above in the hierarchy. For example, the concept labeled by 'A' has $\{1,3,5\}$ as extent, and $\{\mathrm{A},\mathrm{C}\}$ as intent.

An example for an exact rule (implication) which holds in the context is $\{\mathrm{A},\mathrm{B}\} \rightarrow \{\mathrm{C},\mathrm{E}\}$. It can also be read directly in the line diagram: the largest concept having both A and B in its intent is the one labeled by 3 and 5, and it is below or equal to (here the latter is the case) the largest concept having both C and E in its intent. This implication can be derived from two simpler implications, namely $\{\mathrm{A}\} \rightarrow \{\mathrm{C}\}$ and $\{\mathrm{B}\} \rightarrow \{\mathrm{E}\}$. The aim of the frequent Duquenne-Guigues-basis which we introduce in the next section is to provide only a minimal, non-redundant set of implications to the user. That basis will include the two simpler implications.

At the end of this section, we give some simple facts about association rules. We will refer to them later as derivation rules.

Lemma 36 *Rules 1 and 2 hold for* $\phi \in \{\mathrm{conf}, \mathrm{supp}\}$.

1. $\phi(X \rightarrow Y) = \phi(X \rightarrow Y \setminus Z)$, *for all* $Z \subseteq X \subseteq M$, $Y \subseteq M$.

2. $\phi(h(X) \rightarrow h(Y)) = \phi(X \rightarrow Y)$, *for all* $X, Y \subseteq M$.

3. $\mathrm{conf}(X \rightarrow Y) = p \wedge \mathrm{conf}(Y \rightarrow Z) = q \Rightarrow \mathrm{conf}(X \rightarrow Z) = p \cdot q$, *for all frequent concept intents* $X \subset Y \subset Z$.

3'. $\mathrm{supp}(X \rightarrow Z) = \mathrm{supp}(Y \rightarrow Z)$, *for all* $X, Y \subseteq Z$.

4. $\mathrm{conf}(X \rightarrow X) = 1$, *for all* $X \subseteq M$.

Proof The proofs for the confidence are given in [24].

1. $\mathrm{supp}(X \rightarrow Y) = \mathrm{supp}(X \rightarrow Y \setminus Z)$ follows from $X \cup Y = X \cup (Y \setminus Z)$ and the definition of the support count.

2. $\mathrm{supp}(h(X) \rightarrow h(Y)) = \mathrm{supp}(X \rightarrow Y)$ follows from $g(h(X) \cup h(Y)) = g(h(X)) \cap g(h(Y)) = g(f(g(X))) \cap g(f(g(Y))) = g(X) \cap g(Y) = g(X \cup Y)$ by using the facts $g(f(g(X))) = g(X)$ and $g(X \cup Y) = g(X) \cap g(Y)$ provided in [16].

3'. $\mathrm{supp}(X \rightarrow Z) = \frac{|g(X \cup Z)|}{|G|} = \frac{|g(Z)|}{|G|} = \frac{|g(Y \cup Z)|}{|G|} = \mathrm{supp}(Y \rightarrow Z)$ □

□

5.6.2 Bases for Association Rules

In this section, we recall the definition of *iceberg concept lattices* and show that one can derive all frequent itemsets and association rules from them. Then we characterize the *Duquenne-Guigues basis for exact association rules* and the *Luxenburger basis for approximate association rules* and show that all other association rules can be derived from these two bases.

Definition 38 A concept (O, I) is called *frequent concept* if $\mathrm{supp}(I)\ (= \frac{|O|}{|G|}) \geq$ *minsupp*. The set of all frequent concepts is called *iceberg concept lattice*. An itemset I is called *frequent intent* (or *frequent closed itemset*) if it is intent of a frequent concept (i.e., its support is at least *minsupp*). The set of all frequent closed itemsets in $\mathbb{K}$ is denoted FC. ◊

Frequent closed itemset	support
$\emptyset$	1.0
{C}	0.8
{AC}	0.6
{BE}	0.8
{BCE}	0.6
{ABCE}	0.4

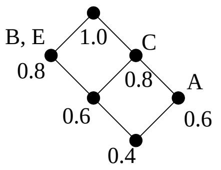


Figure 5.7: Frequent closed itemsets extracted from $\mathbb{K}$ for *minsupp* = 0.4.

Example 8 The frequent closed itemsets in the context $\mathbb{K}$ for *minsupp*=0.4 are presented in Figure 5.7 together with the semi-lattice of all frequent concepts. Both the table and the diagram provide the same information. Note that, in general, the set of frequent concepts is not a lattice, but only a semi-lattice (consider e. g. *minsupp*= 0.5 in the example).

Lemma 37 ([32]) *i) The support of an itemset I is equal to the support of the smallest closed itemset containing I, i. e., $supp(I) = supp(h(I))$.*

ii) The set of maximal frequent itemsets $\{I \in FI \mid \nexists I' \in FI : I \subset I'\}$ is identical to the set of maximal frequent closed itemsets $\{I \in FC \mid \nexists I' \in FC : I \subset I'\}$.

The next theorem shows that the set of frequent closed itemsets with their support is a small collection of frequent itemsets from which all frequent itemsets with their support and all association rules can be derived. I. e., it is a condensed representation in the sense of Mannila and Toivonen [25]. This theorem follows from Lemma 37.

Theorem 38 *All frequent itemsets and their support, as well as all association rules holding in the dataset, their support, and their confidence can be derived from the set FC of frequent closed itemsets with their support.*

Duquenne-Guigues Basis for Exact Association Rules

Next we present the Duquenne-Guigues basis for exact association rules. It is based on the following closure operator.

Theorem 39 *The set $FI \cup \{M\}$ is a closure system on M, and its related closure operator $\overline{\cdot}$ is given by $\overline{I} := h(I)$ if $supp(I) \geq minsupp$ and $\overline{I} := M$ else.*

Proof The set of all frequent itemsets together with M is a closure system, as well as the set of all concept intents. Hence $FI \cup \{M\}$ is, as intersection of those two closure systems, also a closure system. The proof of the fact that $\overline{\cdot}$ is the corresponding closure operator is straightforward. □ □

Our basis adopts the results of [13] to the association rule framework, where additionally the support of the rules has to be considered.

Definition 39 An itemset $I \subseteq M$ in $\mathbb{K}$ is a $\overline{\cdot}$*-pseudo-closed itemset* (or *pseudo-closed itemset* for short) [5] if $\overline{I} \neq I$ and for all pseudo-closed itemsets J with $J \subset I$, we have $\overline{J} \subset I$. The set of all frequent pseudo-closed itemsets in $\mathbb{K}$ is denoted *FP*, the set of all infrequent pseudo-closed itemsets is denoted *IP*. In the (unlikely) case that all itemsets are frequent except the whole set M, we let $IP := \{M\}$ (in order to distinguish this situation from the one where all itemsets are frequent).

The *Duquenne-Guigues basis for exact association rules* (or *frequent Duquenne-Guigues basis*) is defined as the tuple $FDG := (\mathcal{L}, IP)$ with $\mathcal{L} := \{I_1 \to h(I_1) \mid I_1 \in FP\}$ and IP as defined above. ◊

[5] We do not consider pseudo-closed itemsets with respect to other closure operators than $\overline{\cdot}$ (especially not with respect to h) in this paper.

Theorem 40 *From the Duquenne-Guigues basis for exact association rules one can derive all exact association rules holding in the dataset by applying the following rules. Rules ii) to iv) can be applied to $\mathcal{L}$ as long as they do not contradict (i).*

i) If there exists $I \in IP$ with $I \subseteq I_1 \cup I_2$, then $I_1 \rightarrow I_2$ does not hold (because its support is too low).

ii) $X \rightarrow X$ holds.

iii) If $X \rightarrow Z$ holds, then also $X \cup Y \rightarrow Z$.

iv) If $X \rightarrow Y$ and $Y \cup Z \rightarrow W$ hold, then also $X \cup Z \rightarrow W$.

Proof We only sketch the proof here, which applies results of [13] (see also [16]). One has to check that $\mathcal{L} \cup \{I \rightarrow M \mid I \in IP\}$ is the Duquenne-Guigues-basis (in the traditional sense, cf. to [13, 16]) of the closure system $FC \cup \{M\}$. Rule (i) reflects the implications of the form $I \rightarrow M$. □ □

The Duquenne-Guigues basis for exact association rules is not only minimal with respect to set inclusion, but also minimal with respect to the number of rules in $\mathcal{L}$ plus the number of elements in IP, since there can be no complete set with fewer rules than there are frequent pseudo-closed itemsets [13, 16]. Observe that, although it is possible to derive all exact association rules from the Duquenne-Guigues basis, it is not possible in general to determine their support.[6]

Example 9 The set of frequent pseudo-closed itemsets of $\mathbb{K}$ for *minsupp*=0.4 and *minconf*=1/2 is $FP = \{\{A\}, \{B\}, \{E\}\}$, the set of infrequent pseudo-closed itemsets is $IP = \{\{D\}\}$. The Duquenne-Guigues basis is presented in Figure 5.8.

Luxenburger Basis for Approximate Association Rules

In [23, 24], M. Luxenburger discusses bases for partial implications. A *partial implication* is an association rule where the support is not considered. He observed that it is sufficient to consider rules between concept intents only, since $\text{conf}(X \rightarrow Y) = \text{conf}(h(X) \rightarrow h(Y))$. However, his derivation process does not only consist of deduction rules which can be applied in a straightforward manner, but it requires to solve a system of linear equations.

In the KDD process, however, we have to consider the trade-off between the amount of information presented to the user, and the degree of its explicitness. The appearance of the system of linear equations indicates that Luxenburger's results are in favor for a minimal amount of information presented, and against a higher degree of explicitness. As one of the requirements to KDD is that the results should be "ultimately understandable" [14], we want to emphasize more on the explicitness of the results. Therefore we restrict now the expressiveness of the derivation process. This forces the association rules presented to the user to be more explicit.[7]

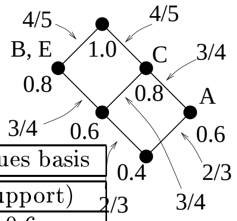
In the sequel, we consider the derivation rules given in Lemma 36. We present a basis for the approximate association rules for these derivation rules.

Definition 40 The *Luxenburger basis for approximate association rules* is given by $LB := \{(r, supp(r), conf(r)) \mid r = I_1 \rightarrow I_2,\ I_1, I_2 \in FC,\ I_1 \prec I_2,\ conf(r) \geq minconf,\ supp(I_2) \geq minsupp\}$. ◊

[6]Even if the support for all rules in the basis is known. With the knowledge about all frequent closed itemsets and their support however, this is possible (see Theorem 38).

[7]Note that in the KDD setting the user will never actually perform longer series of inference steps.

Luxenburger basis		
Approximate rule	Support	Confidence
BCE → A	0.4	2/3
AC → BE	0.4	2/3
BE → C	0.6	3/4
C → BE	0.6	3/4
C → A	0.6	3/4
∅ → BE	0.8	4/5
∅ → C	0.8	4/5



Duquenne-Guigues basis	
$\mathcal{L}$	(Support)
A → C	0.6
B → E	0.8
E → B	0.8
$IP = \{\{D\}\}$	

Figure 5.8: Duquenne-Guigues and Luxenburger bases for *minsupp*=0.4 and *minconf*=1/2.

Theorem 41 *From the Luxenburger basis LB for approximate association rules one can derive all association rules holding in the dataset together with their support and their confidence by using the rules given in Lemma 36. Furthermore, LB is minimal (with respect to set inclusion) with this property.*

Proof In order to determine if an association rule $r := I \to J$ holds in a context (and for determining its support and its confidence) one can consider the rule $I' \to J'$ with $I' := h(I)$ and $J' := h(I \cup J)$ which has (by Rules 1 & 2) the same support and the same confidence. If $I' = J'$, then $conf(r) = 1$ and $supp(r) = \mathrm{supp}(I')$. If $I' \neq J'$, then exists a path of approximate rules, i.e., there are frequent closed itemsets $I_1, \ldots, I_n$ with $I_i \to I_{i+1} \in LB$ and $I' = I_1$ and $I_n = J'$. Support and confidence of r can now be determined by $supp(r) = supp(I_n)$ (Rule 3') and $conf(r) = \Pi_{i=1}^{n-1} conf(I_i \to I_{i+1})$ (Rule 3).

Now we show the minimality of LB. Let $r := I \to J \in LB$. We show that the confidence of r cannot be derived from $LB \setminus \{r\}$ by applying the rules of Lemma 2. Rule 1 cannot be applied forward since J already contains I. It cannot be applied backward because of $I \prec J$. Rule 2 cannot be applied forward since $I = h(I)$ and $J = h(J)$. It cannot be applied backward as LB contains only rules with closed antecedent and closed consequent. Rule 3 cannot be applied since there is no $K \subset M$ with $I \to K \in LB \setminus \{r\}$ and $K \to J \in LB \setminus \{r\}$ (because of $I \prec J$). Rule 4 cannot be applied since $I \neq J$. □ □

Remark. A basis in the sense of [24] is a maximal spanning tree of our basis (when considered as *undirected* graph) containing at most one rule with M as conclusion.[8]

Example 10 The Luxenburger basis for approximate association rules of $\mathbb{K}$ for *minsupp*=0.4 and *minconf*=1/2 is also presented in Figure 5.8. It provides the same information as the list in Figure 5.6, but in a more condensed form. The Luxenburger basis is visualized in the line diagram in Figure 5.8: From its definition it is clear, that each approximate rule in the basis corresponds to (at most)[9] one edge in the diagram. The edge is labeled by the confidence of the rule (as a fraction), and its lower vertice is labeled by its support (as a rational). Implications (exact rules) can be read in the diagram in the standard way described in Section 5.6.1.

As example for the proof of Theorem 41, let us check if $\{B\} \to \{A\}$ holds in the context for *minsupp*=0.4 and *minconf*=1/2. We have $I := \{B\}$ and $J := \{A\}$.

[8]The second condition is negligible in KDD, as it follows directly from minsupp $> 0\,\%$.

[9]In general, there may be edges which do not represent any rule in the Luxenburger basis. Consider for instance *minconf* $= 7/10$. In this case, the two lowest edges would not stand for a valid approximate rule, and would hence not be labelled.

The smallest frequent closed itemset containing B is $I' := \{B, E\}$ and the smallest one containing A and B is $J' := \{A, B, C, E\}$. In the diagram, I' and J' are always represented by the largest concepts which are below all attributes in I and $I \cup J$, resp. Between the two concepts we find the path $I_1 := I'$, $I_2 := \{B, C, E\}$, and $I_3 := J'$. Hence $\mathit{supp}(B \to A) = \mathit{supp}(J') = 0.4 \geq \mathit{minsupp}$ and $\mathit{conf}(B \to A) = \mathit{conf}(I_1 \to I_2) \cdot \mathit{conf}(I_2 \to I_3) = 3/4 \cdot 2/3 = 2/4 \geq \mathit{minconf}$, which means that the rule holds.

5.6.3 Algorithms for Computing the Bases

The algorithms presented in this paper assume that the iceberg concept lattice is already computed. There are several algorithms for computing iceberg concept lattices: the algorithm Close for strongly correlated data [32], the algorithm A-Close for weakly correlated data [31], the algorithms CLOSET [33], ChARM [44], and TITANIC [39, 40]. The algorithm PASCAL [7] computes all (closed and non-closed) frequent itemsets, but can be upgraded to determine also their closures with almost no additional computation time by using the fact that, for $I \subseteq M$,

$$h(I) = I \cup \{m \in M \setminus I \mid \mathit{supp}(I) = \mathit{supp}(I \cup \{m\})\} \ .$$

When the iceberg concept lattice is computed, then the Duquenne–Guigues basis and finally the Luxenburger basis are computed.

Generating the Duquenne-Guigues basis for Exact Association Rules with Gen-FDG

In this section, we present an algorithm that determines the Duquenne–Guigues basis using the iceberg concept lattice. This algorithm (which has not been presented before) implements Definition 39. As it needs to know the closure of frequent itemsets, it is best applied after an algorithm like PASCAL with the modification mentioned above, ChARM, or CLOSET.

The pseudo-code is given in Algorithm 8. The algorithm takes as input the sets FI_i, $1 \leq i \leq k$, containing the frequent itemsets and their support, and the sets FC_i, $0 \leq i \leq k$, containing the frequent closed itemsets and their support. It first computes the frequent pseudo-closed itemsets iteratively (steps 2 to 17). In steps 2 and 3, the empty set is examined. (It must be either a closed or a pseudo-closed itemset by definition.) The loop from step 4 to 17 is a direct implementation of Definition 39 for the frequent pseudo-closed itemsets. The frequent pseudo-closed i-itemsets, their closure and their support are stored in FP_i. They are used to generate the set $\mathcal{L}$ of implications of the Duquenne-Guigues basis for exact association rules DG (step 18).

The set of infrequent pseudo-closed itemsets is determined in steps 19 to 21 using the function $\mathcal{L}^*$-closure (Algorithm 9). This function uses the fact that, for a given closure system, the set of all closed or pseudo-closed sets forms again a closure system [15]. Hence one can generate all closed sets and pseudo-closed sets iteratively by using the corresponding closure operator $\mathcal{L}^*\text{-closure}(Z) := \bigcup_{i=0}^{\infty} Z_i$ with $Z_0 := Z$ and $Z_{i+1} := Z_i \cup \bigcup\{Y | X \to Y \in \mathcal{L}, X \subset Z_i\}$ [15]. The set $\mathcal{L}$ of implications has the form $\mathcal{L} = \{X_1 \to Y_1, \ldots, X_n \to Y_n\}$.

Generating the Luxenburger Basis for Approximate Association Rules with Gen-LB

The pseudo-code generating the Luxenburger basis for approximate association rules is presented in Algorithm 10. The algorithm takes as input the sets FC_i, $0 \leq i \leq$

Algorithm 8 Generating the Duquenne-Guigues basis with Gen-FDG.

```
1)  $\mathcal{L} \leftarrow \{\}$;
2)  if $(FC_0 = \{\})$ then $FP_0 \leftarrow \emptyset$;
3)  else $FP_0 \leftarrow \{\}$;
4)  for $(i \leftarrow 1;\ i \leq k;\ i{+}{+})$ do begin
5)      $FP_i \leftarrow FI_i \setminus FC_i$;
6)      forall $L \in FP_i$ do begin
7)          $pseudo \leftarrow true$;
8)          forall $P \in FP_j$ with $j < i$ do begin
9)              if $(P \subset L)$ and $(P.\text{closure} \not\subseteq L)$
10)             then do begin
11)                 $pseudo \leftarrow false$;
12)                 $FP_i \leftarrow FP_i \setminus \{L\}$;
13)             endif
14)         end
15)         if $(pseudo = true)$ then $L.\text{closure} \leftarrow \min_{\subseteq}(\{C \in FC_{j>i} \mid L \subseteq C\})$;
16)     end
17) end
18) forall $P \in \bigcup_{i=1}^{n} FP_i$ do $\mathcal{L} \leftarrow \mathcal{L} \cup \{P \rightarrow (P.\text{closure}\setminus P)\}$;
19) $IP \leftarrow \emptyset$;
20) forall $L \in MI$ do $IP \leftarrow IP \cup \{\mathcal{L}^*\text{-closure}(I)\}$;
21) $IP \leftarrow \min_{\subseteq} IP$;
```

Algorithm 9 Function $\mathcal{L}^*$-closure reads X and returns its $\mathcal{L}^*$-closure $\mathcal{L}^*(X)$.

```
1)  $Y \leftarrow X$;
2)  for $(i \leftarrow 1; i = n; i++)$ do $i$.used$\leftarrow$ false;
3)  repeat
4)      changed $\leftarrow$ false;
5)      If Subsets$(IP, Y) \neq \emptyset$ then begin $Y \leftarrow M$; changed$\leftarrow$ true end
6)      else for $(i \leftarrow 1; i \leq n; i++)$ do
7)                      if $X_i \subset Y$ then begin $Y \leftarrow Y \cup Y_i$; changed$\leftarrow$ true end
8)  until not changed;
9)  return Y
```

k, containing the frequent closed itemsets and their support. The output of the algorithm is the Luxenburger basis for approximate association rules LB.

The algorithm iteratively considers all frequent closed itemsets $L \in FC_i$ for $2 \leq i \leq k$. It determines which frequent closed itemsets $L' \in \bigcup_{j<i} FC_j$ are covered by L and generates association rules of the form $L' \rightarrow L \setminus L'$ that have sufficient confidence. During the i^{th} iteration, each itemset L in FC_i is considered (steps 3 to 13). For each set FC_j, $1 \leq j < i$, a set S_j containing all frequent closed j-itemsets in FC_j that are subsets of L is created (step 4). Then, all these subsets of L are considered in decreasing order of their sizes (steps 5 to 12). For each of these subsets $L' \in S_j$, the confidence of the approximate association rule $r := L' \rightarrow L \setminus L'$ is computed (step 7). If the confidence of r is sufficient, r is inserted into LB (step 9) and all subsets L'' of L' are removed from S_l, for $l < j$ (step 10). At the end of the algorithm, the set LB contains all rules of the Luxenburger basis for approximate association rules. The proof of the correctness of the algorithm is given in [28].

Algorithm 10 Generating the Luxenburger basis with Gen-LB.

```
1)  $LB \leftarrow \{\}$;
2)  for ($i \leftarrow 2$; $i \leq k$; $i$++) do begin
3)     forall $L \in FC_i$ do begin
4)        for ($j \leftarrow 0; J < i; j++$) do $S_j \leftarrow$ Subsets($FC_j, L$);
5)        for ($j \leftarrow i-1; J \geq 1; j--$) do begin
6)           forall $L' \in S_j$ do begin
7)              $conf \leftarrow L$.support / $L'$.support;
8)              if ($conf \geq minconf$)
9)                 then $LB \leftarrow LB \cup \{(L' \rightarrow (L \setminus L'), L.\text{support}, conf)\}$;
10)             for ($l \leftarrow j; l \geq 1; l--$) do $S_l \leftarrow S_l \setminus$ Subsets($S_l, L'$);
11)          end
12)       end
13)    end
14) end
```

5.6.4 Experimental Results

We have preformed several experiments on synthetic and real data. The characteristics of the datasets used in the experiments are given in Table 5.1. These datasets are the T10I4D100K synthetic dataset that mimics market basket data,[10] the C20D10K and the C73D10K census datasets from the PUMS sample file,[11] and the Mushrooms dataset describing mushroom characteristics.[12] In all experiments, we attempted to choose significant minimum support and confidence threshold values. We varied these thresholds and, for each couple of values, we analyzed rules extracted in the bases.

Table 5.1: Datasets.

Name	Number of objects	Average size of objects	Number of items
T10I4D100K	100,000	10	1,000
Mushrooms	8,416	23	127
C20D10K	10,000	20	386
C73D10K	10,000	73	2,177

Number of Rules. Table 5.2 compares the size of the Duquenne-Guigues basis for exact rules with the number of all exact association rules, and the size of the Luxenburger basis for approximate rules with the number of all approximate rules. In the case of weakly correlated data (T10I4D100K), no exact rule is generated. The reason is that in such data all frequent itemsets are frequent closed itemsets. However, the Luxenburger basis is relatively small compared to the number of all rules, since only immediate neighbors with respect to the subset order (and not arbitrary pairs of sets) are considered. In the case of strongly correlated data (Mushrooms, C20D10K and C73D10K), the ratio between the size of the bases to the number of all rules which hold is much smaller than in the weekly correlated case, because here only few of the frequent itemsets are closed and have to be considered.

[10]http://www.almaden.ibm.com/cs/quest/syndata.html
[11]ftp://ftp2.cc.ukans.edu/pub/ippr/census/pums/pums90ks.zip
[12]ftp://ftp.ics.uci.edu/~cmerz/mldb.tar.Z

Table 5.2: Number of exact and approximate association rules compared with the number of rules in the Duquenne-Guigues and Luxenburger bases.

Dataset (Minsupp)	Exact rules	D.-G. basis	Minconf	Approximate rules	Luxenburger basis
T10I4D100K (0.5%)	0	0	90%	16,269	3,511
			70%	20,419	4,004
			50%	21,686	4,191
			30%	22,952	4,519
Mushrooms (30%)	7,476	69	90%	12,911	563
			70%	37,671	968
			50%	56,703	1,169
			30%	71,412	1,260
C20D10K (50%)	2,277	11	90%	36,012	1,379
			70%	89,601	1,948
			50%	116,791	1,948
			30%	116,791	1,948
C73D10K (90%)	52,035	15	95%	1,606,726	4,052
			90%	2,053,896	4,089
			85%	2,053,936	4,089
			80%	2,053,936	4,089

Relative Performance. Our experiments also show that in all cases the execution time of Gen-FDG and Gen-LB are insignificantly small compared to those of the computation of the iceberg concept lattice, since both algorithms need not access the database. We can conclude that without additional computation time (compared to other approaches, like e. g. Apriori) our approach not only computes all frequent closed itemsets but also the two bases described in Section 5.6.1.

5.7 Bibliographic Notes

Association rule mining with Formal Concept Analysis is a relatively new approach. Other approaches addressing the problem of reducing large sets of association rules provide users with mechanisms for filtering rules, for instance by user defined templates [5, 22], Boolean [27, 36] or SQL-like [26] operators or by introducing further measures of "usefulness" [9]; or they attempt to minimize the number of extracted rules a priori by using information about taxonomies [18, 20, 35] or by applying statistical measures like Pearson's correlation or the χ^2-test [11]. All these approaches have in common that they lose some information.

In [6], we have presented another pair of bases which are different from those presented here. They provide rules with minimal antecedents and maximal consequents. Compared to the results presented here, they have the disadvantage of a higher total number of rules. For the approximate rules, M. Zaki has presented similar results in [45].

———(to be written)———