Constrained Cube Lattices for Multidimensional Database Mining

Lotfi Lakhal, Aix-Marseille Université, France
Alain Casali, Aix-Marseille Université, France
Rosine Cicchetti, Aix-Marseille Université, France
Sébastien Nedjar, Aix-Marseille Université, France

ABSTRACT

In multidimensional database mining, constrained multidimensional patterns differ from the well-known frequent patterns from both conceptual and logical points of view because of a common structure and the ability to support various types of constraints. Classical data mining techniques are based on the power set lattice of binary attribute values and, even adapted, are not suitable when addressing the discovery of constrained multidimensional patterns. In this paper, the authors propose a foundation for various multidimensional database mining problems by introducing a new algebraic structure called cube lattice, which characterizes the search space to be explored. This paper takes into consideration monotone and/or anti-monotone constraints enforced when mining multidimensional patterns. The authors propose condensed representations of the constrained cube lattice, which is a convex space, and present a generalized levelwise algorithm for computing them. Additionally, the authors consider the formalization of existing data cubes, and the discovery of frequent multidimensional patterns, while introducing a perfect concise representation from which any solution provided with its conjunction, disjunction and negation frequencies. Finally, emphasis on advantages of the cube lattice when compared to the power set lattice of binary attributes in multidimensional database mining are placed.

Keywords: Datacubes, Inclusion-Exclusion Identities, Lattices, Multidimensional Database Mining, Search Space.

INTRODUCTION AND MOTIVATIONS

The extraction of constrained multidimensional patterns in the area of multidimensional database mining (OLAP database mining or data mining from categorical database relations) is achieved for solving various problems such as discovering multidimensional association rules (Lu, Feng, & Han, 2000), Roll-Up dependencies (Calder, Ng, & Wijsen, 2002), multidimensional constrained gradients (Dong et al., 2004), closed constrained gradients (Wang, Han, & Pei, 2006), classification rules (Liu, Hsu, & Ma,
1998; Li, Han, & Pei, 2001), correlation rules (Brin, Motwani, & Silverstein, 1997; Grahe, Lakshmanan, & Wang, 2000), datacube (Gray et al., 1997; Beyer & Ramakrishnan, 1999; Han, Pei, Dong, & Wang, 2001; Xin, Han, Li, & Wang, 2003), iceberg cube (Beyer & Ramakrishnan, 1999; Han et al., 2001), emerging cube (Nedjar, Casali, Cicchetti, & Lakhal, 2007; Casali, Nedjar, Cicchetti, & Lakhal, 2009b), quotient cube (Lakshmanan, Pei, & Han, 2002), and closed cube (Casali, Cicchetti, & Lakhal, 2003b; Li & Wang, 2005; Xin, Shao, Han, & Liu, 2006; Casali, Nedjar, Cicchetti, & Lakhal, 2009a). We believe that a precise semantics is required for characterizing the search space and solving these multidimensional data mining problems. Such semantics can be captured through an algebraic structure, the cube lattice, provided with a similar expression power than the power set lattice, which is used for binary database mining (transaction database mining (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996)).

Adapting to this new multidimensional context, approaches and algorithms successfully used when mining binary databases (and thus using the power set lattice as a search space) is possible but not relevant. However, such adaptations have been frequently proposed for the extraction of quantitative association rules (Srikant & Agrawal, 1996) and for classification (Liu et al., 1998; Li et al., 2001). Moreover, (Beyer & Ramakrishnan, 1999; Han et al., 2001) have extended Apriori (Agrawal et al., 1996) for computing iceberg cubes and observed that such extensions “perform terribly”. Reasons behind these failures are the following. Firstly each multidimensional attribute must be replaced by a set of binary attributes, each of which representing a single value of the multidimensional attribute (Srikant & Agrawal, 1996). If the original attribute domains are large (which is the case in data warehouses (Beyer & Ramakrishnan, 1999)), the attributes substitution results in dealing with a great number of binary attributes (called items). On the other hand, the search space, considered when mining binary databases, is the lattice representing the power set of items. Nevertheless, this large search space encompasses a great number of solutions, which are known to be semantically erroneous in a multidimensional context. Let us suppose that in a relation the attribute A has k distinct values. It is replaced by k binary attributes a₁,..., aₖ. In the powerset lattice, all the couples (aᵢ,aⱼ) with l ≤ i, j < k and i ≠ j are considered and evaluating a constraint requires a costly memory space and a pass over the binary relation. But we know that the original data set does not contain any pattern (aᵢ,aⱼ), simply because the initial attribute A is atomic and thus its values aᵢ and aⱼ are exclusive.

If an anti-monotone constraint w.r.t. inclusion (e.g., Frequency(≥ threshold) is enforced for mining frequent multidimensional patterns, the original complexity of levelwise algorithms is altered because the negative border (fundamental concept for the complexity analysis (Mannila & Toivonen, 1997)) used for pruning is enlarged with a possibly voluminous set of useless combinations. The problem worsens when a monotone constraint (e.g., Frequency(≤ threshold) is considered because, although erroneous, the pattern (aᵢ,aⱼ) belongs to the yielded results.

In this paper, we make the following contributions.

1. We introduce and characterize the search space to be considered for multidimensional database mining problems. Such a search space only encompasses semantically valid solutions. By introducing an order relation between elements of this space, and proposing two construction operators, we define a new algebraic structure, called cube lattice. In such groundwork, the extraction of multidimensional patterns can be achieved by using conjunctions of monotone and/or anti-monotone constraints according to our order relation.

2. We formalize condensed representations of constrained cube lattices. Condensed representations based on boundary sets (or borders) avoid to enumerate all the solutions (Mannila & Toivonen, 1996, 1997).
Their practical advantage is that they limit the memory explosion problem especially critical when mining constrained multidimensional patterns. Moreover, condensed representations make it possible to build up the whole solution space or without performing such a construction to decide whether such or such multidimensional pattern is a solution or not. We show that the constrained cube lattice is a convex space and thus it can be represented through boundaries.

3. We show that the cube lattice is graded and describe a generalized levelwise algorithm without “backtracking” devised to mine borders of constrained cube lattices.

4. We use the constrained cube lattice structure to formalize different cubes: the datacube (Gray et al., 1997), the iceberg cube (Beyer & Ramakrishnan, 1999), the range cube, the differential cube (Casali, 2004) and the emerging cube (Nedjar et al., 2007; Casali et al., 2009b).

5. When considering frequent multidimensional patterns, an additional contribution is the proposal of a perfect concise representation of frequent multidimensional patterns. It has two original features: on one hand it is not based on the power set lattice framework and on the other hand it differs from the representations using closed patterns (Pasquier, Bastide, Taouil, & Lakhal, 1999; Pei, Han, & Mao, 2000; Stumme, Taouil, Bastide, Pasquier, & Lakhal, 2002; Zaki & Hsiao, 2005) and non derivable patterns (Calders & Goethals, 2007). From such a representation, we show that any frequent multidimensional patterns (along with its conjunction, disjunction and negation frequencies) are obtained using an improved version of inclusion-exclusion identities. Experimental results are performed in order to compare the size of the essential frequent tuples, the size of iceberg cubes and the one of iceberg closed cubes.

6. According to our knowledge, it not exist a specific algebraic approach for extracting various kinds of multidimensional patterns; therefore we propose a comparison between our approach and extensions to the multidimensional context of approaches mining binary databases. We show in particular: (i) the relevance of our search and solution spaces when compared to the ones considered by the quoted extensions, and (ii) the preservation of levelwise algorithm complexity in our approach and its alteration for the considered extensions.

**Organization of the Paper**

In the second section, we detail the structure of the cube lattice. In the third section, we study its condensed representations for the various cases of constraint conjunctions. The algorithm computing such representations is given in the fourth section. In the fifth section, we use our framework for characterizing various types of cubes (datacube, iceberg cube ...). Assuming that we are dealing with frequent multidimensional patterns, our perfect concise representation of the solution space is defined in the sixth section. We propose a comparison between the cube lattice and the power set lattice in the seventh section. As a conclusion, we underline the advantages of our proposal and evoke further work. Proofs are given in appendix.

This article consolidates research work presented in the conference papers (Casali, Cicchetti, & Lakhal, 2003a, 2005; Casali, Nejar, Cicchetti, & Lakhal, 2007).

**THE CUBE LATTICE FRAMEWORK**

In contrast with patterns considered when mining binary databases, multidimensional patterns are provided with a common structure and their characterization must exhibit such a structure. On another hand, links existing between patterns capture an important semantics. When addressing the former issue, we propose the concept of multidimensional space. Its elements represent
multidimensional patterns. We soundly define links between such patterns through an order relation and propose two basis construction operators. Then the search space, for the various multidimensional database mining problems previously evoked, is characterized by defining the concept of cube lattice.

**Multidimensional Spaces**

Throughout the paper, we make the following assumptions and use the introduced notations. Let be a relation over the schema R. Attributes of R are divided into two sets (i) D the set of dimensions, also called categorical or nominal attributes, which correspond to analysis criteria for OLAP database, classification or concept learning (Mitchell, 1997) and (ii) M the set of measures (for OLAP) or class attributes. Moreover, attributes of D are totally ordered (the underlying order is denoted by <) and \( \forall A \in D, r(A) \) stands for the projection of \( r \) over A.

The multidimensional space of the categorical database relation \( r \) groups all the valid combinations built up by considering the value sets of attributes in D, which are enriched with the symbolic value ALL (also denoted by \( \ast \ast \)). The latter, introduced in (Gray et al., 1997) when defining the operator Cube-By, is a generalization of all the possible values for any dimension.

**Definition 1 (Multidimensional Space).** The multidimensional space of \( r \) is noted and defined as follows: Space(\( r \)) = \{ \times_{A \in D} r(A) \cup \{ \text{ALL} \} \cup \{(\text{\texttt{\_\_\_\_\_\_\_\_\_\_\_\_}})\} \) where \( \times \) symbolizes the Cartesian product, and \( (\text{\texttt{\_\_\_\_\_\_\_\_\_\_\_\_}}) \) stands for the combination of empty values. Any combination belonging to the multidimensional space is a tuple and represents a multidimensional pattern.

**Example 1.** Table 1 presents the categorical database relation used all along the paper to illustrate the introduced concepts. In this relation, A, B, C are dimensions and M is a measure.

The following tuples \( t_1 = (a_1, b_1, \text{ALL}), t_2 = (a_2, b_1, c_1), t_3 = (a_1, b_2, c_1), t_4 = (a_1, \text{ALL}, c_1) \) and \( t_5 = (\text{ALL}, b_1, \text{ALL}) \) are elements of Space(\( r \)).

**Generalization Order**

The tuples of the multidimensional space capture information at various granularity levels. In fact, any tuple of the original relation is generally involved in the construction of several tuples (at the first level of data synthesis, one of the dimensional attribute is provided with the value ALL); the latter tuples can in turn be synthesized and so on. At the most general level, the synthesis consists of a single tuple only encompassing ALL values. It provides the most compact summary of \( r \) but also the roughest. The multidimensional space of \( r \) is structured by the generalization/specialization order between tuples. This order is originally introduced by Mitchell (1997) in the context of machine learning. In a data warehouse context,

<table>
<thead>
<tr>
<th>Row</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>a1</td>
<td>b1</td>
<td>c2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>a1</td>
<td>b2</td>
<td>c1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>a1</td>
<td>b2</td>
<td>c2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>a2</td>
<td>b1</td>
<td>c1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>a2</td>
<td>b1</td>
<td>c1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 1. Relation example r*

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
this order has the same semantic as the operator Rollup/Drilldown (Lakshmanan et al., 2002).

**Definition 2** (Generalization relation). Let \( u, v \) be two tuples of the multidimensional space of \( r \):

\[
u \leq_s v \iff \begin{cases} \forall A \in D \text{ such that } u[A] = v[A] \\ \text{or } v = (\emptyset, \ldots, \emptyset)\end{cases}
\]

If \( u \leq_s v \), we say that \( u \) is more general than \( v \) in \( \text{Space}(r) \).

The covering relation of \( \leq_s \) is noted \( \leq_g \) and defined as follows: \( \forall t, t' \in \text{Space}(r), t \leq_g t' \iff t \leq_s t' \) and \( \exists t' \in \text{Space}(r) \text{ such that } t <_g t' \).

The direct lower bound of a tuple \( t \), noted \( \text{DLB}(t) \), encompasses all the tuples which are covered by \( t \) (\( \text{DLB}(t) = \{ t' \in \text{Space}(r) \text{ such that } t' \leq_g t \} \)).

**Example 2.** In the multidimensional space of our relation example (cf. Table 1), we have:

\([\text{ALL}, b_2, \text{ALL}] \leq_g (a_1, b_1, c_1)\), i.e., \([\text{ALL}, b_2, \text{ALL}]\) is more general than \((a_1, b_1, c_1)\) and \((a_1, b_2, c_1)\) is more specific than \((\text{ALL}, b_1, \text{ALL})\) and \((a_1, b_1, \text{ALL})\) \(\leq_g (a_1, b_2, c_1)\).

Moreover any tuple generalizes the tuple \((\emptyset, \ldots, \emptyset)\) and specializes the tuple \((\text{ALL}, \text{ALL}, \text{ALL})\).

When applied to a set of tuples, the \( \min \) and \( \max \) operators yield the tuples that are the most general ones in the set or the most specific ones respectively.

**Definition 3** (min/max operators). Let \( T \subseteq \text{Space}(r) \) be a set of tuples:

\[
\begin{align*}
\min_{\leq_g}(T) &= \{ t \in T \text{ such that } \not\exists u \in T : u \leq_g t \}, \\
\max_{\leq_g}(T) &= \{ t \in T \text{ such that } \not\exists u \in T : t \leq_g u \}.
\end{align*}
\]

**Example 3.** In our multidimensional space, let us consider \( T = \{(a_1, b_1, \text{ALL}), (a_1, b_1, c_1), (a_1, \text{ALL}, c_1)\} \). Thus we have \( \min(T) = \{ (a_1, b_1, \text{ALL}), (a_1, \text{ALL}, c_1) \} \) and \( \max(T) = \{ (a_1, b_1, c_1) \} \).

**Basis Operators**

The two basic operators provided for tuple construction are Sum (denoted by \( + \)) and Product (noted \( \ast \)). The Sum of two tuples yields the most specific tuple that generalizes the two operands.

**Definition 4** (Sum operator). Let \( u \) and \( v \) be two tuples in \( \text{Space}(r) \).

\[
t = u + v \iff \forall A \in D, t[A] = \begin{cases} u[A] \text{ if } u[A] = v[A] \\
\text{ALL otherwise.}\end{cases}
\]

We say that \( t \) is the Sum of the tuples \( u \) and \( v \).

**Example 4.** In our example of \( \text{Space}(r) \), we have \((a_1, b_1, c_1) + (a_1, b_2, c_1) = (a_1, \text{ALL}, c_1)\). This means that \((a_1, \text{ALL}, c_1)\) is built up from the tuples \((a_1, b_1, c_1)\) and \((a_1, b_2, c_1)\), and it generalizes both of them.

The Product of two tuples yields the most general tuple which specializes the two operands. If, for two tuples, there exists a dimension \( A \) having distinct and real world values (i.e. existing in the original relation), then the only tuple specializing them is the tuple \((\emptyset, \ldots, \emptyset)\).

Apart from it, the sets of tuples that can be used to retrieve them are disjoined.

**Definition 5** (Product operator). Let \( u \) and \( v \) be two tuples in \( \text{Space}(r) \). Then,
\[
\begin{align*}
t = (\varnothing, \ldots, \varnothing) & \text{ if } \exists A \in D,
\text{ such that } u[A] \neq v[A] \neq \text{ALL}, \\
t & = u[A] \text{ if } v[A] = \text{ALL}, \\
t & = \text{ALL} \text{ if } u[A] = v[A], \\
\text{otherwise } \forall A \in D,
\end{align*}
\]

We say that \( t \) is the Product of the tuples \( u \) and \( v \).

**Example 5.** In our example of \( \text{Space}(r) \), we have \((a,b,\text{ALL}) \cdot (a,c,\text{ALL},c) = (a,b,c,c)\). This means that \((a,b,\text{ALL})\) and \((a,c,\text{ALL},c)\) generalize \((a,b,c,c)\) and \((a,b,c,c)\) participates to the construction of \((a,b,\text{ALL})\) and \((a,c,\text{ALL},c)\) (directly or not). The tuples \((a,b,\text{ALL})\) and \((a,c,\text{ALL},c)\) have no common point apart from the tuple of empty values.

In order to characterize a chief feature of our search space, we need to know the attributes for which the values associated with a tuple \( t \) are different from the value \( \text{ALL} \). This is why the function \( \text{Attribute} \) is introduced.

**Definition 6 (Attribute function).** Let \( t \) be a tuple of \( \text{Space}(r) \), we have \( \text{Attribute}(t) = \{A \in D \mid t[A] = \text{ALL}\} \).

**Example 6.** In our example of \( \text{Space}(r) \), we have \( \text{Attribute}((a,b,\text{ALL})) = \{A, B\} \).

**Characterization of the Cube Lattice**

By providing the multidimensional space of \( r \) with the generalization order between tuples and using the above-defined operators \text{Sum} and \text{Product}, we define an algebraic structure which is called Cube Lattice. Such a structure provides a sound foundation for multidimensional database mining issues. We give the fundamental properties of the cube lattice which are resumed in theorem 1. In this section, we make use of concepts well known when dealing with lattices, thus we adopt the conventional notations in the domain (Gantner & Wille, 1999), and state their equivalence with ours.

**Lemma 1.** The ordered set \( \text{CL}(r) = (\text{Space}(r), \leq) \) is a complete lattice and has the following properties:

\[
\forall T \subseteq \text{CL}(r), \; \land T = \bigwedge_{t \in T} t \text{ where } \land \text{ stands for the infimum.}
\]

\[
\forall T \subseteq \text{CL}(r), \; \lor T = \bigvee_{t \in T} t \text{ where } \lor \text{ symbolizes the supremum.}
\]

Through the following proposition, we characterize the order embedding from the cube lattice towards the power set lattice of the whole set of attribute values. To avoid ambiguities, each value is prefixed by the name of the concerned attribute.

**Proposition 1.** Let \( \text{PL}(r) \) be the power set lattice of attribute value set, i.e. the lattice

\[
\mathcal{P}\left(\bigcup_{A \in D} A, \forall a \in r(A), \subseteq\right).
\]

Then it exists an order-embedding \( \varphi : \text{CL}(r) \rightarrow \text{PL}(r) \)

\[
t \mapsto \begin{cases} 
\bigcup_{A \in D} A.a, \forall a \in r(A) & \text{if } t = (\varnothing, \varnothing, \varnothing) \\
\{A.t[A], \forall A \in \text{Attribute}(t)\} & \text{otherwise.}
\end{cases}
\]

Consequently, if \( t \leq t' \) then \( \varphi(t) \subseteq \varphi(t') \).

The rank of a tuple \( t \) is the length of the minimal path which links it to the tuple \( (\text{ALL}, \ldots, \text{ALL}) \).
Thus we have: \( \text{rank}(t) = \begin{cases} |D| + 1 & \text{if } t = (\mathcal{E}, \leq, \boxempty) \\ \Phi(t) \text{ otherwise} \end{cases} \) 

\( \forall T \subseteq \text{CL}(r), \wedge T = \bigwedge_{t \in T} t \)

\( \forall T \subseteq \text{CL}(r), \bigvee T = \cdot_{t \in T} t \)

**Reminder:** The coatoms (the atoms respectively) are the maximal elements, i.e. the most specific tuples (the minimal elements respectively, i.e., the most general tuples), of the lattice deprived of its universal upper bound: \( T \) (its lower bound respectively: \( \bot \)). We denote by:

1. \( \text{At}(\text{CL}(r)) \) the atoms of the cube lattice (i.e., \( \{ t \in \text{CL}(r) : |\Phi(t)| = 1 \} \)),
2. \( \text{Cl}(\text{CL}(r)) \) the coatoms of the cube lattice (i.e., \( \{ t \in \text{CL}(r) : |\Phi(t)| = |D| \} \)),
3. \( \text{At}(t) = \{ t' \in \text{At}(\text{CL}(r)) \text{ such that } t' \leq t \} \) the atoms of a tuple \( t \).

**Lemma 2.** \( \text{CL}(r) \) is a coatomistic lattice.

**Lemma 3.** \( \text{CL}(r) \) is an atomistic lattice.

**Theorem 1.** Let \( r \) be a categorical database relation over \( D \cup M \). The ordered set \( \text{CL}(r) = (\text{Space}(r), \subseteq) \) is a complete, atomistic and coatomistic lattice, called cube lattice in which Meet (\( \wedge \)), or GLB, and Join (\( \vee \)), or LUB, operators are given by:

**Figure 1. Hasse diagram of the cube lattice of \( r \)**
The total number of elements in the cube lattice is:

\[(\prod_{A \in \mathcal{D}_I} (| r(A) | + 1)) + 1\]

**CONDENSED REPRESENTATIONS OF CONSTRAINED CUBE LATTICES**

The cube lattice defines a graded search space for various multidimensional database mining problems. In this section, we study the structure of the cube lattice in presence of constraint conjunctions. Provided with such a structure, we propose condensed representations (or borders) of the constrained cube lattice with a twofold objective: defining in a compact way the solution space and deciding whether a tuple \( t \) belongs to the solution space or not. Finally, following from principles of levelwise approaches, we show that the cube lattice is graded and give an algorithm for computing condensed representations of constrained cube lattices. We take into account monotone and anti-monotone constraints frequently used in binary data mining (Ross, Srivastava, Stuckey, & Sudarshan, 1998; Raedt & Zimmermann, 2007; Pei, Han, & Lakshmanan, 2004). These constraints can be applied to:

1. Measures of interest (Bayardo & Agrawal, 1999) such as frequency of patterns, confidence, correlation, entropy... In these cases only the dimensional attributes of \( R \) are necessary;
2. Additive statistical functions (e.g., COUNT, SUM, MIN, ...) which apply to measures of \( M \) for computing aggregates;
3. Class prediction measures for supervised classification approaches, in that case, apart from the dimensions of \( D, R \) encompasses class attributes (also noted \( M \)). We recall the definitions of monotone and anti-monotone constraints w.r.t. \( \leq_g \).

**Definition 7** (Constraint).

1. A constraint \( \text{Const} \) is monotone if and only if: \( \forall \ t, u \in \text{CL}(r): a. [t \leq u \text{ and Const}(t)] \Rightarrow \text{Const}(u). \)
2. A constraint \( \text{Const} \) is anti-monotone if and only if: \( \forall \ t, u \in \text{CL}(r): b. [t \leq u \text{ and Const}(u)] \Rightarrow \text{Const}(t). \)

**Example 8.** In the multidimensional space example, we would like to know all the tuples for which the value of the sum for the attribute measure is greater or equal to 3. The constraint “SUM\(_{\text{val}}(M) \geq 3\)” is an anti-monotone constraint. In the same way, if we intend to mine all the tuples from which the sum of the values for the attribute \( M \) is less or equal to 5, the expressed constraint “SUM\(_{\text{val}}(M) \leq 5\)” is monotone.

**Structure of the Constrained Cube Lattice**

The cube lattice faced with monotone and/or anti-monotone constraints does not necessarily remain a lattice. We show in this section that such a partially ordered set is provided with a mathematical structure which is a convex space (Vel, 1993) and thus it can be represented by it boundary (Hirsh, 1991).

**Notations:** We note \( \text{cmc} \) (\( \text{camc} \) respectively) a conjunction of monotone constraints (anti-monotone respectively) and \( \text{chc} \) an hybrid conjunction of constraints (monotone and anti-monotone). According to the considered case, the introduced borders are indexed by the kind of the considered constraint. For example, \( U_{\text{cmc}} \) symbolizes the upper border for the anti-monotone constraints (i.e. the set of the most specific tuples that satisfy the conjunction of anti-monotone constraints).
Remarks (extreme cases):

1. We suppose that the tuple \((ALL,\ldots,ALL)\)
always satisfies the conjunction of an anti-monotone constraints and the tuple \((\varnothing,\ldots,\varnothing)\) always verifies the conjunction of monotone constraints. Under these assumptions, the solution space encompasses at least one element (possibly the tuple of empty values).

2. Moreover, we assume that the tuple \((ALL,\ldots,ALL)\) never verifies the conjunction of monotone constraints and the tuple \((\varnothing,\ldots,\varnothing)\) never satisfies the conjunction of antimonotone constraints, because without making these assumptions, the solution space is \(CL(r)\).

**Definition 8** (Convex Space). Let \((P,\leq)\) be a partially ordered set, \(C \subseteq P\) is a convex space if \(\forall x, y, z \in P, x \leq y \leq z\) and \(x, z \in C \Rightarrow y \in C\). Thus, \(C\) is bounded by two sets: an upper bound or “upper set” defined by \(U = \max_i(C)\) and a lower bound or “lower set” defined by \(L = \min_i(C)\).

**Theorem 2.** The constrained cube lattice \(CL(r)\) is a convex space. Its upper set \(U\) and lower set \(L\) are:

- If \(\text{const} = \text{cme}\), \(L_{\text{cme}} = \min\{t \in CL(r) | \text{cme}(t)\}\) and \(U_{\text{cme}} = (\varnothing,\ldots,\varnothing)\).

- If \(\text{const} = \text{cme}\), \(L_{\text{cme}} = (\varnothing,\ldots,\varnothing)\) and \(U_{\text{cme}} = \max\{t \in CL(r) | \text{cme}(t)\}\).

The generic upper bound \(U_{\text{const}}\) represents the most specific tuples satisfying the constraint conjunction and the lower bound \(L_{\text{const}}\) is the set of the most general tuples satisfying the constraint conjunction. Thus, \(L_{\text{cme}}\) and \(U_{\text{cme}}\) are condensed representations of the constrained cube lattice with conjunction of monotone and/or anti-monotone constraints.

**Corollary 1.**

Given \(U_{\text{cme}}\) and \(L_{\text{cme}}\), the condensed representation of \(CL(r)\) is:

- \(L_{\text{cme}} = \min\{t \in CL(r) | \exists t' \in U_{\text{cme}} : t \leq t' \text{ and } \text{cme}(t')\}\)

- \(U_{\text{cme}} = \{t \in U_{\text{cme}} | \exists t' \in L_{\text{cme}} : t' \leq t\}\).

Given \(L_{\text{cme}}\) and \(U_{\text{cme}}\), the condensed representation of \(CL(r)\) is:

- \(L_{\text{cme}} = \max\{t \in CL(r) | \exists t' \in L_{\text{cme}} : t \leq t'\}\)

- \(U_{\text{cme}} = \{t \in U_{\text{cme}} | \exists t' \in U_{\text{cme}} : t \leq t'\}\).

**Example 9.** Table 2 gives the borders \(U_{\text{cme}}, L_{\text{cme}}, U_{\text{che}}, L_{\text{che}}\) of the cube lattice of the relation example by considering the hybrid constraint “\(3 \leq \text{SUM}_{val}(M) \leq 5\)”.

<table>
<thead>
<tr>
<th>(U_{\text{cme}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((a, b, c))</td>
</tr>
<tr>
<td>((a, ALL, c))</td>
</tr>
<tr>
<td>((a, b, ALL))</td>
</tr>
<tr>
<td>((a, ALL, c))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(L_{\text{cme}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((ALL, b, ALL))</td>
</tr>
<tr>
<td>((a, b, c))</td>
</tr>
<tr>
<td>((a, ALL, c))</td>
</tr>
<tr>
<td>((ALL, ALL, c))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(U_{\text{che}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((a, b, c))</td>
</tr>
<tr>
<td>((a, ALL, c))</td>
</tr>
<tr>
<td>((a, b, ALL))</td>
</tr>
<tr>
<td>((a, c))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(L_{\text{che}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((ALL, b, ALL))</td>
</tr>
<tr>
<td>((a, b, c))</td>
</tr>
<tr>
<td>((a, ALL, c))</td>
</tr>
<tr>
<td>((ALL, ALL, c))</td>
</tr>
</tbody>
</table>

---

Table 2: Borders for the cube lattice constrained by “\(3 \leq \text{SUM}_{val}(M) \leq 5\)”

---

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
The Algorithm GLA

Levelwise algorithms are proved to be efficient when dealing with very large data sets stored on disk and when the underlying search space is a graded lattice. Improvements of these algorithms have been proposed in order to minimize the number of database scans and optimize the candidate generation step (Bastide, Taouil, Pasquier, Stumme, & Lakhal, 2000; Geerts, Goethals, & Bussche, 2001; Stumme et al., 2002). This is why we adopt their principles when proposing an algorithm which computes borders of constrained cube lattices. More precisely, the algorithm GLA yields the borders $U$ and $L$ corresponding to the type of considered constraints and uses whether $L^-$ (i.e., the negative border of $CL(r)_{c_{\text{max}}}^-$) or $L$ for the pruning step.

GLA follows from general principles of levelwise algorithms and introduces the following particular features:

i. The level construction step (or candidate generation) makes use of the operator Sub-Product, defined below using the operator Product,

ii. The pruning step is performed without “back tracking” like in (Dehaspe & Toivonen, 1999), and

iii. Monotone and anti-monotone constraints as well as their conjunction are taken into consideration.

Contrarily to a pattern language (Mannila & Toivonen, 1997; Raedt, Jaeger, Lee, & Mannila, 2002) which is a general model, the cube lattice is a specific pattern language for multidimensional database mining. The advantage is that the cube lattice is graded which is a fundamental property for convex spaces (cf., theorem 2) and for using levelwise algorithms. Obviously such a fundamental property is difficult to be proved in a general model.

**Proposition 3.** The cube lattice $CL(r)$ is graded.

**Definition 9** (Sub-Product operator). Let $u$ and $v$ be two tuples of $Space(r)$, $X = \text{Attribute}(u)$ and $Y = \text{Attribute}(v)$.

$$
t = u \odot v \iff \begin{cases} t = u \cdot v & \text{if } X \setminus \max_{\leq D} (X) \\
= Y \setminus \max_{\leq D} (Y) \text{ and } \\
\max_{\leq D} (X) <_{D} \max_{\leq D} (Y) \\
(\emptyset, \ldots, \emptyset) & \text{otherwise.} \end{cases}
$$

where $<_\cdot$ is a total order over $D$. Let us notice that $|\Phi(u)| = |\Phi(v)|$ is a necessary condition for the relevance of the Sub-Product operator.

The Sub-Product operator is a constrained product operator useful for candidate generation in a levelwise approach (Agrawal et al., 1996; Mannila & Toivonen, 1997). Provided with multidimensional patterns at the level $i$, the Sub-Product operator generates candidates of level $i + 1$, if they exist (else the Sub-Product yields $\emptyset, \ldots, \emptyset$). Moreover, each tuple is generated only once.

**Example 10.** In our example, we have $(a_1,b_1,ALL) \leq_{\emptyset} (a_1, ALL, c_1) = (a_1,b_1,c_1)$ but $(a_1, ALL, c_1) \leq_{\emptyset} (a_1,b_1,ALL) = (\emptyset, \ldots, \emptyset)$.

**Complexity of the GLA Algorithm**

Mannila and Toivonen (1997) have studied complexity analysis for levelwise algorithms by using the positive and negative borders. The following complexity is given in (Gunopulos, Mannila, Khardon, & Toivonen, 1997):

$$O((|BD| + |Sol|) \times \text{cost of camc test})$$

where $BD$ is the negative border (e.g., infrequent minimal elements in the context of frequent pattern discovery, $L^-$ in our case) and $Sol$ is the solution set of the problem. Such a complexity can be easily generalized for the levelwise algorithm GLA which deals with monotone and/or anti-monotone constraint conjunctions. The complexity of GLA is: $O((|P| + |Q|) \times \max(\text{cost of camc test, cost of cmc test}))$ where:

$$\text{If } \text{camc} = \{t\} : P = L_{\text{camc}} \text{ and } Q = \{t \in CL(r) \text{ such that } \neg \text{camc}(t)\}.$$
Algorithm 1 - GLA Algorithm

Input: relation r over R, camec and cmec 
Output: U, L

if cmec = {} then L := {{ALL, ..., ALL}}
else L := A(At(r)) 
\[ L := \{ t \in \text{At}(r) | \text{cmec}(t) \text{ and camec}(t) \} \] 
L_{i+1} := L_i \cup L 
end if

if camec = {} then U := {{\varnothing, ..., \varnothing}}
else U := L_i := \{ t \in \text{At}(r) | \text{cmec}(t) \} 
\[ L := \{ t \in \text{At}(r) | \text{cmec}(t) \} \] 
end if

while (L = \varnothing) do
if (camec = {}) then
\[ C_{i+1} := \{ v = t \oplus t' | t, t' \neq (\varnothing, ..., \varnothing) \text{ and } \exists u \in L : u \leq v \} \]
\[ L := \{ t \in C_{i+1} | \text{cmec}(t) \} \]
L_{i+1} := L_i \cup L 
end if
else
\[ C_{i+1} := \{ v = t \oplus t' | t, t' \in L, v \neq (\varnothing, ..., \varnothing) \text{ and } \exists u \in L : u \leq v \} \]
\[ L := \{ t \in C_{i+1} | \text{cmec}(t) \} \]
L_{i+1} := L_i \cup L 
end if
if cmec = {} then L := \min_{\leq} (L \cup \{ t \in L_i | \text{cmec}(t) \})
i := i + 1
end while
U := \{ t \in U | \exists t' \in L : t' \leq_{gL} t \}
return U, L

Else P = L_{\text{camec}} and Q = CL(r)_{\text{camec}}

FORMALIZATION OF EXISTING CUBES

In this section, we review different variants of datacubes and, by using the constrained cube lattice structure, propose a characterization both simple and well founded.

Datacubes

Originally proposed by Gray et al. (1997), the datacube according to a set of dimensions is presented as the result of all the Group By which can be expressed using a combination of dimensions. The result of any Group By is called a cuboid, and the set of all the cuboids is structured within a relation noted Datacube(r). The schema of this relation remains similar to the one of r, i.e., D U M and the very same schema is used for all the cuboids (in order to perform their union) by enforcing a simple idea: any dimension which is not involved in the computation of a cuboid (i.e., not mentioned in the Group By) is provided with the value ALL. For any attribute set X ⊆ D, a cuboid of the datacube, noted Cuboid(X, f(\{M*\})), is yielded by the following SQL query:

```
Select [ All, ] X, f(\{M*\})
From r
Group By X;
```

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Thus a datacube can be achieved by the one of two SQL queries:

1. By using the operator Cube By (or Group By Cube according to the DBMS):

   ```sql
   Select D, f(\{M\}*)
   From r
   Cube By D;
   ```

2. By performing the union of all the cuboids:

   ```latex
   \text{Datacube}(r, f(\{M \mid \ast\}))) = \bigcup_{X \subseteq D} \text{Cuboid}(X, f(\{M \mid \ast\}))
   ```

   This union is expressed in SQL as follows:

   ```sql
   Select f(\{M\}*)
   From r
   Union
   Select A, f(\{M\}*)
   ```

   **Example 11.** In our example, the set of all the aggregative queries can be expressed by using the operator Cube as follows:

   ```sql
   Select A, B, C, Sum(M)
   From r
   Cube By A, B, C;
   ```

   The previous query results in $2^3=8$ cuboids: according to ABC, AB, AC, BC, A, B, C and $\emptyset$. The cuboid according to ABC corresponds to the original relation itself. A tuple $t$ belongs to the datacube of $r$ if and only if it exists a tuple $t'$ in $r$ which specializes $t$; else $t$ cannot be built up. As a consequence, whatever the aggregative function is, the tuples of the datacube projected over the selected

---

**Figure 2. Datacube of $r$ w.r.t. COUNT (\"\*\"
$\leftrightarrow$ ALL)**

---

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
dimensions remain invariant, only the values computed by the aggregative function vary.

**Proposition 4.** Let \( r \) be a relation projected over \( D \), the set of tuples (i.e., without the measure values) of the datacube of \( r \) is a convex cube for the constraint “\( \text{Count}_{\text{sql}}(t) \geq 1 \)”: 

\[
\text{Datacube}(r) = \{ t \in CL(r) \mid \text{Count}_{\text{sql}}(t) \geq 1 \}
\]

Since the constraint “\( \text{Count}_{\text{sql}}(t) \geq 1 \)” is an antimonotone constraint (according to \( \leq \)), a datacube is a convex cube. By applying theorem 2, we infer that any datacube can be represented by two borders: the relation \( r \) which is the upper set and the tuple \( (\text{ALL}, \ldots, \text{ALL}) \) which is the lower set. Then we can easily assess the appurtenance of any tuple \( t \) to the datacube of \( r \); we have just to find a tuple \( t' \in r \) specializing \( t \).

**Example 12.** Figure 2 exemplifies the datacube of our relation example (Cf. Table 1) using the function COUNT. The measure associated to a tuple is written after it (e.g., \( (a_1, \text{ALL}, c_4) \)). For example, in this diagram, the tuples of the cuboid according to AC are in bold.

In this section, we have shown that we can characterize the datacube as a convex cube. In the similar way, in the following section we take advantage of the genericity of our structure to capture various types of cubes.

**Others Cubes**

Like the datacube, most of the existing cubes can achieved by SQL queries or by using our structure. Hereafter, we present the cubes the most used in practice.

- Inspired from frequent patterns, Beyer et al. introduce the Iceberg cubes (Beyer & Ramakrishnan, 1999) which are presented as tuple subset of the datacube satisfying for the measure values a minimal threshold constraint. The proposal is motivated by the following objective: the decision makers are interested in general tendencies, the relevant trends are trends sufficiently distinctive. Thus it is not necessary to compute and materialize the whole cube (the search space is pruned). This results in a significant gain for both execution time and required storage space.

- Windows cubes gather tuples having measure values which fit in a given range. Such cubes place emphasis on middle tendencies, not too general and not too specific.

- Differential cubes (Casali, 2004) result from the set difference between the datacubes of two relations \( r_1 \) and \( r_2 \). They capture tuples relevant in a relation and not existing in the other. In contrast with the previous ones, such cubes capture comparisons between two data sets. For instance in a distributed application, these data sets can be issued from two different sites and their differential cube highlights trends which are common here and unknown there. For OLAP applications as well as data stream analysis, trend comparisons along time are strongly required in order to exhibit trends, which are significant at a moment and then disappear or on the contrary non-existent trends, which later appear in a clear-cut way. If we consider that the original relation \( r_1 \) is stored in a data warehouse and \( r_2 \) is made of refreshment data, the differential cube shows what is new or dead.

- Emerging cubes (Nedjar et al., 2007; Casali et al., 2009) capture trends, which are not relevant for the users (because under a threshold) but which grow significant or on the contrary general trends which soften but not necessarily disappear. Emergent cubes enlarge results of differential cubes and refine cube comparisons. They are of particular interest for data stream analysis because they exhibit trend reversals. For instance, in a web application where
Concise Representations of Frequent Multidimensional Patterns

Lattice

In this section, we propose a perfect concise representation of the frequent cube lattices. From such a representation, any frequent multidimensional pattern along with its frequency is derived by using an improved version of inclusion-exclusion identities adapted to the cube framework.

Frequency Measures and Inclusion Exclusion Identities

Let us consider \( t \in \mathbb{CL}(r) \) (\( r \) is a relation over \( D \cup M \)), we define three weight measures, which are compatible with the weight functions defined in (Stumme et al., 2002), for \( t; (i) \) its frequency (noted \( \text{Freq}(t) \)), (ii) the frequency of its disjunction (noted \( \text{Freq}(\vee t) \)) and (iii) the frequency of its negation (noted \( \text{Freq}(\neg t) \)).

\[
\text{Freq}(t) = \frac{\sum_{t' \in [M]} t'[M] \text{ such that } t \leq t'}{\sum_{t' \in [M]} t'[M]} 
\]

(1)

\[
\text{Freq}(\vee t) = \frac{\sum_{t' \in [M]} t'[M] \text{ such that } t + t' = (ALL, \ldots, ALL)}{\sum_{t' \in [M]} t'[M]} 
\]

(2)

\[
\text{Freq}(\neg t) = \frac{\sum_{t' \in [M]} t'[M] \text{ such that } t + t' = (ALL, \ldots, ALL)}{\sum_{t' \in [M]} t'[M]} 
\]

(3)

Example 13. In the cube lattice of our relation example, we have:

- \( \text{Freq}((a_1,b_1, ALL)) = 5/11 \), \( \text{Freq}(\vee (a_1,b_1, ALL)) = 1 \) and \( \text{Freq}(\neg (a_1,b_1, ALL)) = 0 \).
- The inclusion-exclusion identities make it possible to state, for a tuple \( t \), the relationship between its frequency, the frequency of its disjunction and the frequency of its negation, as follows:

\[
\text{Freq}(\vee t) = \sum_{t' \leq t} (-1)^{|\Phi(t')|-1} \text{Freq}(t') 
\]

(4)

\[
\text{Freq}(t) = \sum_{t' \leq t} (-1)^{|\Phi(t')|-1} \text{Freq}(\vee t') 
\]

(5)

\[
\text{Freq}(\neg t) = 1 - \text{Freq}(\vee t) 
\]

(6)

Where \( \Phi \) is the order-embedding (Cf. proposition 1) and \( \perp \) stands for the tuple \((ALL, \ldots, ALL)\).

Example 14. In the cube lattice of our relation example, we have:

- \( \text{Freq}((a_1, b_1, ALL)) = \text{Freq}((a_1, ALL, ALL)) + \text{Freq}((ALL, b_1, ALL)) - \text{Freq}((a_1, b_1, ALL)) = 9/11 + 7/11 - 5/11 = 1 \)
- \( \text{Freq}((a_1, b_1, ALL)) = \text{Freq}((a_1, ALL, ALL)) + \text{Freq}((ALL, b_1, ALL)) - \text{Freq}(\vee (a_1, b_1, ALL)) = 9/11 + 7/11 - 1 = 5/11 \)
- \( \text{Freq}(\neg (a_1, b_1, ALL)) = 1 - \text{Freq}(\vee (a_1, b_1, ALL)) = 0 \).

Computing the frequency of the disjunction for a tuple can be performed along with computing its frequency and thus the execution time of levelwise algorithms is not altered.

Provided with the frequency of the disjunction for the tuples, a concise representa-
tion of frequent tuples can be defined and the computation of the negation frequency is straightforward.

**Frequent Essential Tuples**

**Definition 10** (Essential tuple). Let \( t \in CL(r) \) be a tuple, and \( \minfreq \) a given threshold. We say that \( t \) is frequent if and only if \( \text{Freq}(t) \geq \minfreq \). If \( t = (\text{ALL}, \ldots, \text{ALL}) \), then \( t \) is said essential if and only if:

\[
\text{Freq}(t) = \max_{t' \in \text{Essential}(r)} (\text{Freq}(t'))
\]  

(7)

where \( \text{DLB}(t) = \{t \in CL(r) \mid t' \leq_{g} t \} \). Let us note \( \text{Essential}(r) \) the set of essential tuples.

**Example 15.** In the cube lattice of our relation example (Cf. Table 1), \((a, b, \text{ALL})\) is an essential tuple because \( \text{Freq}(\bigvee (a, b, \text{ALL})) \neq \text{Freq}(\bigvee (a, \text{ALL}, \text{ALL}, \text{ALL})) \) and \( \text{Freq}(\bigvee (a, b, \text{ALL})) \neq \text{Freq}(\bigvee (\text{ALL}, \text{ALL}, \text{ALL})) \).

**Lemma 4.** Let us consider the twofold constraint: “\( t \) is frequent” (C1) and “\( t \) is an essential tuple” (C2). Such a constraint conjunction is anti-monotone according to the order \( g \) (i.e., if \( t' \) is either an essential or a frequent tuple and \( t^* \leq_{g} t \), then \( t^* \) is also essential or frequent).

**Frequency Inference Using Improved Inclusion Exclusion Identities**

The following formulas show firstly how to compute the frequency of the disjunction from the set of essential tuples and secondly how to optimize the inclusion-exclusion identities for finding efficiently the frequency of a frequent tuple. A naive method for computing the frequency of a tuple \( t \) requires the knowledge of the disjunctive frequencies of all the tuples, which are more general than \( t \). Formula 8, is an intuitive optimization based on the essential tuple concept while formula 9 is an original derivation of the frequency of \( t \).

**Lemma 5.**

\[
\forall t \in CL(r), \text{Freq}(t) = \max_{t' \in \text{Essential}(r)} (\text{Freq}(t') \mid t' \leq_{g} t).
\]

**Theorem 3.** \( \forall t \in CL(r), t \notin \text{Essential}(r) \), let \( u = \text{Argmax}(\{\text{Freq}(t') \mid t' \leq_{g} t \text{ and } t' \in \text{Essential}(r)\}) \), then we have:

\[
\text{Freq}(t) = \sum_{t' \leq_{g} t} (-1)^{|t| - |t'|} \text{Freq}(u) \text{ if } u \leq_{g} t
\]

\[
\text{Freq}(t') \text{ otherwise. (8)}
\]

**Perfect Concise Representation of Frequent Multidimensional Patterns**

**Definition 11** (Perfect Concise Representation). \( \text{CR}(r) \) is a Concise Representation for the set of frequent tuples, noted \( \text{FCL}(r) \), if and only if (i) \( \text{CR}(r) \subseteq \text{FCL}(r) \) and (ii) \( \forall t \in \text{FCL}(r) \), its frequency can be derived from \( \text{CR}(r) \).

Unfortunately, the set \( \text{Essential}(r) \) cannot be a concise representation of \( \text{FCL}(r) \), this is why we add the set of maximal frequent tuples \( (U_{Ci}) \) to obtain a perfect concise representation. Thus a tuple is frequent if and only if it generalizes a tuple of \( U_{Ci} \).

**Theorem 4.** \( U_{Ci} \cup \{t \in CL(r) \mid t \text{ is a frequent essential tuple}\} \) is a perfect concise representation of the frequent cube lattice.
Algorithm 2 - GLAE Algorithm

\[ U_{C_1} \coloneqq \text{Max Set Algorithm}(r, C_1) \]
\[ L_1 = \{t \in \text{At}(CL(r)) \mid \exists u \in U_{C_1} : t \leq u \} \]

while \( L_i \neq \varnothing \) do

\[ C_{i+1} := \{v = t \odot t \mid t, t' \in L_i, v \neq (\varnothing, \ldots, \varnothing), \exists u \in U_{C_i} : v \leq u \text{ and } \forall w \in DLB(v), w \in L_i} \]

Scan the database to find the frequencies of the disjunction \( \forall t \in C_{i+1} \)

\[ L_{i+1} := C_{i+1} \setminus \{t \in C_{i+1} | \exists t' \in DLB(t) : \text{Freq}(\forall t) = \text{Freq}(\forall t')\} \]

\[ i := i + 1 \]

end while

return \( \bigcup_{j=1}^{i} L_j \)

---

Table 3. Essential tuple for “\( \text{Freq}(t) \geq 2/11 \)”

<table>
<thead>
<tr>
<th>rank</th>
<th>Tuple</th>
<th>Freq(( \forall t ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a(_r), ALL, ALL)</td>
<td>9/11</td>
</tr>
<tr>
<td>1</td>
<td>(ALL, b(_1), ALL)</td>
<td>7/11</td>
</tr>
<tr>
<td>1</td>
<td>(ALL, b(_2), ALL)</td>
<td>4/11</td>
</tr>
<tr>
<td>1</td>
<td>(ALL, ALL, c(_i))</td>
<td>7/11</td>
</tr>
<tr>
<td>1</td>
<td>(ALL, ALL, c(_j))</td>
<td>4/11</td>
</tr>
<tr>
<td>2</td>
<td>(a(_r), b(_1), ALL)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(a(_r), ALL, c(_i))</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(ALL, b(_1), c(_i))</td>
<td>9/11</td>
</tr>
<tr>
<td>2</td>
<td>(ALL, b(_2), c(_i))</td>
<td>9/11</td>
</tr>
<tr>
<td>2</td>
<td>(ALL, b(_2), c(_j))</td>
<td>6/11</td>
</tr>
</tbody>
</table>

---

Claims:

1. In an OLAP database context, when the function \( \text{COUNT} \) is used, we have: \( \text{Datacube}(r) = \{(t, \text{freq}(t)) \mid t \in CL(r) \text{ and } \text{Freq}(t) > 0\} \). Thus, frequent essential tuples and \( U_{C_i} \) provide a perfect concise representation of data cubes.

2. The framework of frequent essential tuples can be used in the context of frequent pattern mining and the set of frequent essential patterns is a perfect concise representation of frequent patterns.

---

The Algorithm GLAE

In order to yield the frequent essential tuples, we propose a levelwise algorithm with maximal frequent pattern \( U_{C_i} \) pruning. Our algorithm includes the function Max Set Algorithm which discovers maximal frequent multidimensional patterns. It could be enforced by modifying the algorithm Max-Miner (Bayardo, 1998).

Example 16. The concise representation of the relation illustrated in Table 1 for the antimonotone constraint “\( \text{Freq}(t) \geq 2/11 \)” is the following: the set of frequent essential
tuples is exemplified in Table 3 and the set $U_{C1}$ is given in Table 2.

We aim to know if $(a,b,c)$ is frequent and if it is, what is the frequency of its disjunction and negation.
- Since $(a, b, c) \subseteq (a, b, c, e)$, the former tuple is frequent. We use theorem 3 to retrieve its frequency: $\text{Freq}(a, b, c) = \text{Freq}(a, ALL) + \text{Freq}(b, ALL) = \text{Freq}(\bigvee (a, b, c, e)) = \text{Freq}(a, ALL, e) = 4/11$
- In order to retrieve the frequency of the disjunction, we apply lemma 5: $\text{Freq}(\bigvee (a, b, c, e)) = \text{Freq}(\bigvee (a, b, c)) = 9/11$.
- We use De Morgan law to retrieve the frequency of the negation: $\text{Freq}^-(a, b, c) = 1 - \text{Freq}(\bigvee (a, b, c)) = 2/11$.

Let us retrieve the frequencies of the tuple $(a, b, c)$.
- Since $(a, b, c) \subseteq U_{C1}$, it is frequent. We use theorem 3 to compute its frequency: $\text{Freq}(a, b, c) = \text{Freq}(a, ALL) + \text{Freq}(b, ALL) + \text{Freq}(c, ALL)$.
- We use lemma 5 to yield the frequency of the disjunction: $\text{Freq}(\bigvee (a, b, c, e)) = \text{Freq}(\bigvee (a, b, c)) = 1$.
- We use De Morgan law to retrieve the frequency of the negation: $\text{Freq}(\neg (a, b, c)) = 1 - \text{Freq}(\bigvee (a, b, c)) = 0$.

We would like to know if $(a, b, c)$ is frequent.

Since $\not\exists t \in U_{C1}$, $(a, b, c, e)$ is frequent.

Experimental Evaluations

By providing the disjunctive and the negative frequencies, the proposed approach enriches the results obtained with the two other perfect covers proposed in the literature. Our objective is now to show, through various experiments, that the size of this new cover is often smaller than the size of the cover based on the frequent closed patterns and this in the most critical cases: strongly correlated data. For meeting this objective, we evaluate the number of frequent essential tuples and compare it with the number of frequent closed tuples (iceberg closed cube (Casal et al., 2009a)) and the size of frequent tuples (iceberg cube) by using five datasets. Those datasets can be found at http://fimi.cs.helsinki.fi/. The characteristics of the datasets used for experiments are given in Table 4. They are:

- The dataset Chess,
- The dataset Connect,
- The dataset Mushrooms describing the characteristics of mushrooms,
- The datasets of census Pumsb and Pumsb*, extracted from "PUMS sample file". Pumsb* is the same dataset than Pumsb from which are removed all the patterns which have a threshold greater or equal to

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of transactions</th>
<th>Average size of each transaction</th>
<th>Number of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess</td>
<td>3196</td>
<td>37</td>
<td>75</td>
</tr>
<tr>
<td>Connect</td>
<td>65557</td>
<td>43</td>
<td>129</td>
</tr>
<tr>
<td>Mushrooms</td>
<td>8124</td>
<td>23</td>
<td>119</td>
</tr>
<tr>
<td>Pumsb</td>
<td>49046</td>
<td>74</td>
<td>2113</td>
</tr>
<tr>
<td>Pumsb*</td>
<td>49046</td>
<td>50.5</td>
<td>2088</td>
</tr>
</tbody>
</table>

Table 4. Dataset
80%, the synthetic datasets T10I4D100K and T20I6D100K, built from sale data.

For all the experiments, we choose relevant minimum thresholds.

In these five datasets, only encompassing strongly correlated data, the ratio between frequent patterns and the total number of patterns is high. Thus we are in the most difficult cases. For finding the positive border we use Max-Miner (Bayardo, 1998).

In the dataset Pumsb*, using either the frequent closed patterns or the frequent essential patterns as a cover is advantageous: the gain compared to the set of frequent patterns for the dataset Pumsb* with the threshold Minfreq = 20% is about 45. On the other hand, for this dataset, even if the approach by essential patterns is better than the one with closed patterns, the obtained gain is near to one. In the three remaining datasets, the approach by essential is very efficient.

With the dataset Chess, many of frequent patterns are closed patterns, but the number of essential patterns is relatively small. This results in a benefit, for the threshold Minfreq = 50%, of a factor 40 compared to the original approach and of a factor 20 compared to the approach using closed patterns. With the dataset Connect and a threshold Minfreq = 70%, the benefit compared to frequent patterns is approxi
Figure 4. Experimental results for connect

Table 5. Differences between the cube lattice of $r$ and the associated powerset lattice

<table>
<thead>
<tr>
<th></th>
<th>Cube Lattice CL(r)</th>
<th>Power set Lattice PL(r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order relation</td>
<td>Generalization($\leq$)</td>
<td>Inclusion ($\subseteq$)</td>
</tr>
<tr>
<td>Meet Operator</td>
<td>Sum (+)</td>
<td>Intersection ($\cap$)</td>
</tr>
<tr>
<td>Join Operator</td>
<td>Product ($\cdot$)</td>
<td>Union ($\cup$)</td>
</tr>
<tr>
<td>Size of largest level</td>
<td>$O\left(\frac{2^{</td>
<td>I</td>
</tr>
</tbody>
</table>
mately of a factor 2500 and compared to closed frequent pattern of a factor 20. We can see that with the dataset Pumsb, the benefit compared to the approach by frequent closed patterns is of a factor 20 for a threshold $\text{Minfreq} = 60\%$ and compared to the approach by frequent patterns is approximately 40.

For more readability in the figures, we have omitted “frequent patterns” in the legends. Thus “simple” means frequent tuples, “closed” stands for frequent closed tuples and “essential” symbolizes frequent essential tuples (Figure 3, Figure 4, Figure 5, Figure 6, Figure 7).

**COMPARISON WITH RELATED WORK**

According to our knowledge it does not exist a specific approach aiming to state an algebraic foundation for multidimensional or OLAP database mining. Several algorithms attempt to extend, to the multidimensional context, proposals successfully used for mining binary databases. They use the power set lattice as the search space. In (Mannila & Toivonen, 1997; Raedt et al., 2002; Raedt & Zimmermann, 2007), a pattern language approach is introduced as a general theoretical model for data mining.
Unfortunately, this attempt of formalization has a drawback: the property of graduation is not always preserved whereas it is fundamental for levelwise algorithms and to state a convex structure. This is why we believe that each mining context requires a specific approach: power set lattice for the binary databases and cube lattice for multidimensional databases.

We propose in this section a comparative analysis between binary data mining approaches, like quantitative association rules (Srikant & Agrawal, 1996) and classification rules, used in a multidimensional context and ours. We perform this comparison by studying search spaces to be traversed, solution spaces, and the behavior of levelwise algorithms. By considering the power set lattice $PL(r)$ and the cube lattice $CL(r)$ as search spaces for the discovery of constrained multidimensional patterns, our comparison focuses on four points: lattice and level sizes, lattice characteristics, correctness of obtained solutions when faced with constraint conjunctions, and levelwise
algorithm complexity. Table 5 summarizes the differences between the two lattices.

**Lattice and Level Sizes**

Let us examine the size of the compared lattices and their largest level. The following contributes to a more precise analysis of the computational complexity of multidimensional database mining algorithms. \( | \text{PL}(r) | = 2^{|\sum A \in D \text{r}(A)} | \), whereas \( | \text{CL}(r) | = \prod_{A \in D} (| \text{r}(A) | + 1) + 1 \) (Cf. 2). An upper bound for the cube lattice cardinality is \( \Theta((\max_{A \in D}(| \text{r}(A) | + 1))^{\dim D}) \). Let us consider for instance a relation with 5 attributes, each of which having 10 possible values, we have \( | \text{PL}(r) | = 2^{50} \), whereas \( | \text{CL}(r) | = 11^5 + 1 \).

We set \( n = \sum_{A \in D} | \text{r}(A) | \). The size of the largest level in \( \text{PL}(r) \) is bounded by \( \left( \frac{n}{\sqrt{\log n}} \right) \), which is asymptotic to \( \frac{n}{\sqrt{\log n}} \sqrt{\log \log n} \) whereas the maximal size of levels in the cube lattice is bounded by \( \left( \frac{|D|}{\dim D/2} \right) \max_{A \in D}(| \text{r}(A) |)^{|D|} \) which is asymptotic to \( \frac{n^{|D|/2}}{\sqrt{|D|} \sqrt{\log \log n}} \max_{A \in D}(| \text{r}(A) |)^{|D|} \).
Thus the size of the largest level in PL(r) is exponential in the value number of dimensional attributes of the relation (i.e., n). On the other hand, the size of CL(r) is exponential in the number of attributes (i.e., |D|).

**Lattice Characteristics**

Since the order relation of the two considered lattices is different, we can deduce two consequences:

- $\land$ and $\lor$ operators are different in the two lattices: in the power set lattice $\land = \cap$ and $\lor = \cup$, whereas in the cube lattice $\land = +$ and $\lor = -$.

- The power set lattice and the cube lattice are two complete, atomistic, co-atomistic and graded lattices but the cube lattice is not distributive.

**Solution Correctness**

The power set lattice PL(r) encompasses solutions semantically erroneous whereas the cube lattice is exactly the valid search space. More precisely, the embedded order $\varphi$ (Cf. proposition 1) shows that for any tuple in the cube lattice it exists an equivalent combination in the power set lattice which is semantically valid whereas the converse equivalence does not hold because $\varphi$ is not bijective. $\forall t \in CL(r)$, $\exists a_i, a_j \in \varphi(t)$ and $a_i, a_j \in r(A_i)$ according to definition 1. On the other hand $\forall A_i \in D$, $\forall a_i, a_j \in r(A_i)$ with $i \neq j$, $(a_i, a_j) \in PL(r)$, nevertheless such combinations are proved to be erroneous because multidimensional patterns cannot encompass two values of the very same attribute. For instance, for the monotone constraint “Freq(ν) = 1”, the set $\{a, b\}$ belongs to the set of solutions if we consider the power set lattice as the search space, but, as previously explained, this set is semantically erroneous. For the anti-monotone constraint “Freq(0) ≤ 2/11”, if we consider the power-set lattice as the search space, the set $\{a, b\}$ belongs to the set of solutions, but it is also erroneous. However, for the anti-monotone constraint “Freq(0) ≥ minfreq” (minfreq is a user threshold), the embedded order $\varphi$ bijective, so we can use either algorithms based on the power set lattice, or algorithms based on the cube lattice in order to find the set of solutions.

**Levelwise Algorithm Complexity**

The generation of erroneous patterns obviously alters performances of the underlying algorithms. We study such an alteration through the comparison of size of borders relevant for monotone (Freq(t) ≤ threshold) and anti-monotone (Freq(t) ≥ threshold) constraints.

Let us consider the most general solutions satisfying cmc for PL(r) and CL(r). We have $|L_{cmc}(PL(r))| \geq |L_{cmc}(CL(r))|$ $+ \sum_{A \in D} \frac{|A|^2 - |A|}{2}$

For anti-monotone constraints, the negative border for PL(r) also encompasses erroneous patterns (couples of values of a very same attribute), its size is greater than the size of $L_{cmc}$ for CL(r). In fact, the number of additional elements in the border $L_{cmc}$ for PL(r) is exactly the maximal number previously given (the very same couples are to be considered), i.e.

$|L_{cmc}(PL(r))| \geq |L_{cmc}(CL(r))|$ $+ \sum_{A \in D} \frac{|A|^2 - |A|}{2}$

The larger the attribute value sets are, the worse are the consequences of the negative border size. This is the reason behind the inefficiency, in a multidimensional context, of levelwise algorithms over PL(r).

**CONCLUSION**

In this paper, we introduce a formal framework for solving various problems of multidimensional database mining. We propose a novel algebraic structure, the cube lattice as a graded search space. We also derived condensed representations of the cube lattice faced with
monotone and/or anti-monotone constraints. Such a result is based on the particular structure of constrained cube lattice, which is a convex space structure. Using this framework, we formalize various existing cubes. We propose an algorithm, which computes borders for the various conjunctions of constraints (w.r.t. generalization) while preserving the original complexity of levelwise algorithms. When considering the discovery of frequent multidimensional patterns, an additional contribution is the proposal of a perfect concise representation of frequent cube lattices. Then, we improve inclusion-exclusion identities for deriving frequencies. Finally, we compare the power set lattice of binary attributes and the cube lattice in order to show that our structure has several advantages: its size, the solution correctness, and the preservation of levelwise algorithm complexity. Defining set operations on constrained cube lattices (convex spaces) is an interesting future work. It could be a basis for providing convex space algebra in the cube lattice framework (with arbitrary monotone and/or anti-monotone constraints given in (Ross et al., 1998; Pei et al., 2004)).

REFERENCES


Lakshmanan, L., Pei, J., & Han, J. (2002). Quotient cube: How to summarize the semantics of a data cube. In *Proceedings of the 28th international conference on very large databases (vldb)* (pp. 778-789).


Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of the 1st international conference on data mining (icdm)* (pp. 369-376).


Pei, J., Han, J., & Mao, R. (2000). CLOSE: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proceedings of the Workshop on research issues in data mining and knowledge discovery (dmkd)* (pp. 21-30).


Lotfi Lakhal received the Phd degree in computer science and the Habilitation for Research Direction from the University of Nice-Sophia-Antipolis (France) respectively in 1986 and in 1991. He is a full professor at the University of Aix-Marseille II - IUT of Aix en Provence and member of the laboratory LIF. His research interest includes Databases, Formal Concept Analysis, Data Mining, Data Warehousing and Multidimensional Skylines.

Alain Casali obtained the Phd degree in computer science from the University of Aix-Marseille (France) in 2005. He is an assistant professor at the University of Aix-Marseille II - IUT of Aix en Provence and is a member of the LIF laboratory. He studies the lattice algorithmic and the multidimensional data mining.

Rosine Cicchetti is a full professor at the University of Aix-Marseille (France) and responsible of the database and machine learning research team at the Laboratory of Fundamental Computer Science (LIF) of Marseilles. She obtained the PhD in 1990 (University of Nice, France) and the Habilitation for Research Direction in 1996 (University of Aix-Marseilles). Her research topics encompass Databases, Data Mining, Data Warehousing, Statitical databases and Multidimensional Skylines.

Sébastien Nedjar obtained the Phd degree in computer science from the University of Aix-Marseille (France) in 2009. His research work concerns OLAP Mining and Data Warehousing and Multidimensional Skylines.
APPENDIX

Proof of lemma 1 – If CL(r) is a lattice then it is complete because it is finite. We show that any couple t, t’ of CL(r) is provided with a lower bound (or infimum) and an upper bound (or supremum).
- We show that CL(r) is a $\land$-semi lattice:
- By definition of the operator $+, t+t' \leq \land t$ and $t+t' \leq \land t'$, thus $t+t'$ is a lower bound for $t$ and $t'$.
- Let us have $u \in \text{Space}(r) | u \leq \land t$ and $u \leq \land t'$, then $u + (t + t') = u + t + t' = u + t'$ (because $u \leq \land t$) = $u$ (because $u \leq \land t'$).
- As a consequence, $u \leq \land t + t'$ and any couples of tuples in Space(r) has an infimum. Thus, CL(r) is a $\land$-semi lattice.

We show now that CL(r) is a $\lor$-semi lattice:
- If $t$ and $t'$ are two tuples such that their product is different from $(\varnothing, \ldots, \varnothing)$ then by definition of the operator $\cdot$, $t \leq \lor t \cdot t'$ and $t' \leq \lor t \cdot t'$. As a consequence $t \cdot t'$ is an upper bound for $t$ and $t'$.
- Let us have $u \in \text{Space}(r) | t \leq \lor u$ and $t' \leq \lor u$ and $t \cdot t' \neq (\varnothing, \ldots, \varnothing)$, then: $u \cdot (t \cdot t') = u \cdot t \cdot t' = u \cdot t'$ (because $t \leq \lor u$) = $u$ (because $t' \leq \lor u$). Thus $t \cdot t' \leq \lor u$ and couples of tuples in Space(r) has a supremum.

Consequently, CL(r) is a $\lor$-semi lattice.

From the two points of the proof, CL(r) is a complete lattice.

Proof of proposition 1 – Obvious because

$$\{A.t[A] | \forall A \in \text{Attribute}(t)\} \subseteq P(\bigcup_{A \in D} A.a, \forall a \in r(A))$$
by construction, and $\varnothing$ is injective.

Proof of lemma 2 - We look for a set $T$ of CL(r) coatoms such that if $t \in CL(r)$, then $t = \land T$. Let us characterize the set of CL(r) coatoms: CAT(CL(r)) = $\times_{A \in D} r(A)$. If $t \in CAT(CL(r))$, then we have just to set $T=\{t\}$. Else, let us consider $t \in CL(r) \setminus CAT(CL(r))$. It exists a set of attributes $X \subseteq D | X=D \setminus \text{Attribute}(t)$.

Let us consider $t_1, t_2 \in \times_{A \in D} r(A) | t \leq \land t_1, t \leq \land t_2$ and $\forall A \in X, t_1[A] \neq t_2[A]$.

By definition of the operator SUM, we have,

$$(t_1 + t_2)[A] = \begin{cases} t_1[A] & \text{if } A \notin X \\ \text{ALL} & \text{otherwise.} \end{cases}$$
Therefore \( t_1 + t_2 = t \). Thus \( T = \{t_1, t_2\} \), and any tuple is the infimum of the coatoms specializing it.

Proof of lemma 3 - We look for a set \( T \) of \( CL(r) \) atoms such that if \( t \in CL(r) \), then \( t = \bigvee T \). If \( t \in \text{At}(CL(r)) \), then \( T = \{t\} \). Otherwise, by definition of the operator Product, if \( t \in CL(r) \setminus \text{At}(CL(r)) \), then \( t = \bigwedge_{t' \in \text{At}(CL(r))} t' \), thus we can set \( T = \{t' \in \text{At}(CL(r)) | t' \leq_{g} t\} \). Moreover according to lemma 1, \( \bigwedge \) is a supremum operator in the lattice \( CL(r) \), thus \( CL(r) \) is an atomistic lattice.

Proof of theorem 1 - The proof of the theorem is directly derived from lemmas 1, 2 and 3.

Proof of theorem 2 -
Let us consider \( CL(r)_{\text{cmc}} = \{t \in CL(r) | \exists u \in U_{\text{cmc}} \text{ and } \exists v \in L_{\text{cmc}} \text{ such that } v \leq_{g} t \leq_{g} u\} \).

We show that \( CL(r)_{\text{cmc}} \) is the set of tuples satisfying the conjunction of monotone constraints. For the need of the demonstration, let us denote \( \text{Sol}_{\text{cmc}} \) this set of solutions. Let \( t \) be a tuple belonging to \( CL(r)_{\text{cmc}} \).

- By definition, it exists \( v \in L_{\text{cmc}} | \text{cmc}(v) \text{ and } v \leq_{g} t \). Since \( \text{cmc} \) is a conjunction of monotone constraints, we have \( \text{cmc}(t) \). Therefore \( t \in \text{Sol}_{\text{cmc}} \). Thus \( CL(r)_{\text{cmc}} \subseteq \text{Sol}_{\text{cmc}} \).

- Let be \( t \in \text{Sol}_{\text{cmc}} \), thus it exists \( v \in L_{\text{cmc}} | v \geq_{g} t \) because \( L_{\text{cmc}} \) represents the set of minimal tuples satisfying \( \text{cmc} \). Moreover, the constraint \( \exists u \in L_{\text{cmc}} | t \geq_{g} u \) is always satisfied. Thus \( t \in CL(r)_{\text{cmc}} \) and \( \text{Sol}_{\text{cmc}} \subseteq CL(r)_{\text{cmc}} \).

From the two above points, we have: \( \text{Sol}_{\text{cmc}} = CL(r)_{\text{cmc}} \).

- true by application of the principle of duality (Ganter & Wille, 1999) over the cube lattice constrained by a conjunction of monotone constraints.

- true because if \( \text{Sol}_{\text{cmc}} = \{t \in CL(r) | \text{ chc}(t)\} \), then \( \text{Sol}_{\text{cmc}} = \text{Sol}_{\text{cmc}} \cap \text{Sol}_{\text{cmc}} \).

Proof of corollary 1 - Obvious from definition of a convex space (Vel, 1993) and the characterization by borders of the constraint cube lattice (cf. theorem 2).

Proof of proposition 3 - \( CL(r) \) is graded iff \( \forall t, t' \in CL(r), t \leq_{g} t' \Rightarrow \text{rank}(t)+1 = \text{rank}(t') \).

\( t \leq_{g} t' \Rightarrow \exists ! A \in D | t[A] = \text{ALL} \text{ and } t'[A] = \text{ALL} \). Thus we have: \( \text{rank}(t)+1 = \text{rank}(t') \).

Proof of proposition 4 - From the definition of a datacube: \( t \in \text{datacube}(r) \Leftrightarrow \exists t' \in r | t \leq_{g} t' \Leftrightarrow \text{Count}_{\text{cul}}(t) \geq 1 \).
Proof of lemma 4 - It is well known that the frequency constraint (Cₙ) is an anti-monotone constraint. Let us focus on the second constraint. For helping us to prove the lemma, we need to introduce two concepts. The first one is the difference between two tuples:

\[ t = u \setminus v \iff \forall A \in D, t[A] = \begin{cases} u[A] & \text{if } u[A] = v[A] \\ \text{ALL otherwise.} & \end{cases} \]

The second one is the function \( g \) which yields, from a cell \( t \), the set of the identifiers of the tuple of the relation which have a common value with \( t \) (i.e. \( g(t) = \{ i' \in \text{Tid}(r) \text{ such that } t' \in r, t'[\text{Tid}]=i \text{ and } t+t'= (\text{ALL, } i', \text{ALL}) \} \)). Since the disjunctive frequency is a monotone increasing function, the function \( g \) is also a monotone increasing function (i.e. \( t \preceq_u t' \Rightarrow g(t) \subseteq g(t') \)). Let \( t \) and \( u \) be two multidimensional tuples such that \( t \preceq_u u \). Let us suppose that \( t \) is not an essential tuple. We show that \( u \) is not an essential tuple too. Since \( t \) is not an essential tuple, it exists \( t' \preceq_u t \) which is an essential tuple. Moreover, we have \( g(t') = g(t) \). We show that \( g(u) = g(u \setminus (t \setminus t')) \).

We know that \( g(t \setminus t') \subseteq g(t) = g(t') \). However, \( g(u) = g((u \setminus (t \setminus t')) \bullet (t \setminus t')) \). By applying the increasing monotony of \( g \) over \((u \setminus (t \setminus t'))\) and \( (t \setminus t') \), we obtain: \( g(u) = g((u \setminus (t \setminus t')) \cup g(t \setminus t')) \)

- Since \( g(t \setminus t') \subseteq g(t') \), we have: \( g(u) \subseteq g((u \setminus (t \setminus t')) \cup g(t') \).

- This expression is re-writen: \( g(u) \subseteq g((u \setminus (t \setminus t')) \bullet t') \). However \( t' \preceq_u u \setminus (t \setminus t') \), we deduce that \( g(u) \subseteq g(u \setminus (t \setminus t')) \).

Since \( u \setminus (t \setminus t') \preceq_u u \), by applying the increasing monotony of \( g \) over those two tuples, we obtain \( g(u \setminus (t \setminus t')) \subseteq g(u) \).

From the two above points, the tuple \( u \) is not an essential tuple since its disjunctive frequency is equal to the one of its generalized cells. As a consequence, the constraint \( t \) is not an essential tuple is a monotone constraint for the generalization order. By applying the principle of duality to this constraint over the cube lattice, we deduce that the constraint \( t \) is an essential tuple is an anti-monotone constraint for the generalization order.

Proof of lemma 5 - If \( t \in \text{Essential}(r) \), then we know its disjunctive frequency. Let us assume that \( t \not\in \text{Essential}(r) \), and \( u = \text{Argmax}(\{ \text{Freq}(\forall t') \mid t' \preceq_u t \text{ and } t' \in \text{Essential}(r) \}) \) (i.e. \( \text{Freq}(\forall u) = \text{Max}(\{ \text{Freq}(\forall t') \mid t' \preceq_u t \text{ and } t' \in \text{Essential}(r) \}) \)). Therefore, we have \( u \preceq_u t \).

Copyright © 2010. IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Moreover the function yielding the frequency of the disjunction is an increasing monotone function, and we have: \( \forall t_i, t_j \in CL(r), t_i \leq_r t_j \Rightarrow \text{Freq}(\bigvee t_i) \leq \text{Freq}(\bigvee t_j) \). Thus applying this formula with \( t_i = u \) and \( t_j = t \) results in: \( \text{Freq}(\bigvee u) \leq \text{Freq}(\bigvee t) \).

If \( \text{Freq}(\bigvee u) < \text{Freq}(\bigvee t) \), then \( t \) is essential (Cf. formula 7), which is contradicting the initial assumption. Thus, \( \text{Freq}(\bigvee u) = \text{Freq}(\bigvee t) \).

Proof of lemma 6 - The proof is based on inclusion-exclusion identities and the fact that \( \forall t' \in [u, t] : \text{Freq}(\bigvee u) = \text{Freq}(\bigvee t') \) (lemma 5).

Proof of theorem 3 - Using lemma 6, we have:

\[
\text{Freq}(t) = \sum_{t' \leq t} (-1)^{|t'| - 1} \text{Freq}(\bigvee t') + \sum_{u \leq t' \leq t} (-1)^{|t'| - 1} \text{Freq}(\bigvee t')
\]

There are \( 2^{\phi(t) - \phi(u)} \) tuples between \( u \) and \( t \) and the frequency of their disjunction is \( \text{Freq}(\bigvee u) \). But, considering a power set lattice, the number of patterns which have an odd-numbered cardinality is equal to the number of patterns which have an even cardinality. Thus the second part of the equation is equal to zero.

Proof of theorem 4 - The proof is based on theorem 2 and on lemmas 4, 5 and 6.