Inductive Databases and Multiple Uses of Frequent Itemsets: The cInQ Approach

Jean-François Boulicaut

Institut National des Sciences Appliquées de Lyon,
LIRIS CNRS FRE 2672, Bâtiment Blaise Pascal
F-69621 Villeurbanne cedex, France

Abstract. Inductive databases (IDBs) have been proposed to afford the problem of knowledge discovery from huge databases. With an IDB the user/analyst performs a set of very different operations on data using a query language, powerful enough to perform all the required elaborations, such as data preprocessing, pattern discovery and pattern post-processing. We present a synthetic view on important concepts that have been studied within the cInQ European project when considering the pattern domain of itemsets. Mining itemsets has been proved useful not only for association rule mining but also feature construction, classification, clustering, etc. We introduce the concepts of pattern domain, evaluation functions, primitive constraints, inductive queries and solvers for itemsets. We focus on simple high-level definitions that enable to forget about technical details that the interested reader will find, among others, in cInQ publications.

1 Introduction

Knowledge Discovery in Databases (KDD) is a complex interactive process which involves many steps that must be done sequentially. In the cInQ project[1], we want to develop a new generation of databases, called "*inductive databases*" (IDBs), suggested by Imielinski and Mannila in [42] and for which a simple formalization has been proposed in [20]. This kind of databases integrate *raw data* with *knowledge* extracted from *raw data*, materialized under the form of patterns into a common framework that supports the knowledge discovery process within a database framework. In this way, the process of KDD consists essentially in a querying process, enabled by a query language that can deal either with raw data or patterns and that can be used throughout the whole KDD process across many different applications. A few query languages can be considered as candidates for inductive databases. For instance, considering the prototypical case of assoc-

[1] This research is partially funded by the Future and Emerging Technologies arm of the IST Programme FET-Open scheme (cInQ project IST-2000-26469). The author warmly acknowledges all the contributors to the cInQ project and more particularly Luc De Raedt, Baptiste Jeudy, Mika Klemettinen, and Rosa Meo.

R. Meo et al.(Eds.): Database Support for Data Mining Applications,LNAI 2682, pp. 1–23, 2004.

iation rule mining, [10] is a comparative evaluation of three proposals (`MSQL` [43], `DMQL` [38], and `MINE RULE` [59]) in the light of the IDBs' requirements.

In this paper, we focus on mining queries, the so-called *inductive queries*, i.e., queries that return patterns from a given database. More precisely, we consider the pattern domain of itemsets and databases that are transactional databases. Doing so, we can provide examples of concepts that have emerged as important within the cInQ project after 18 months of work.

It is useful to abstract the meaning of mining queries. A simple model has been introduced in [55] that considers a data mining process as a sequence of queries over the data but also the so-called *theory* of the data. Given a language $\mathcal{L}$ of patterns (e.g., itemsets, sequences, association rules), the theory of a database r with respect to $\mathcal{L}$ and a selection predicate q is the set $Th(r, \mathcal{L}, q) = \{\phi \in \mathcal{L} \mid q(r, \phi) \text{ is true}\}$. The predicate q indicates whether a pattern ϕ is considered interesting (e.g., ϕ denotes a property that is "frequent" in r). The selection predicate can be defined as a combination (boolean expression) of primitive constraints that have to be satisfied by the patterns. Some of them refer to the "behavior" of a pattern in the data, e.g., its "frequency" in a given data set is above or below a user-given threshold, some others define syntactical restrictions on desired patterns, e.g., its "length" is below a user-given threshold. Preprocessing concerns the definition of the database r, the mining phase is often the computation of the specified theory while post-processing can be considered as a querying activity on a materialized theory or the computation of a new theory.

This formalization however does not reflect the context of many classical data mining processes. Quite often, the user is interested not only in a collection of patterns that satisfy some constraints (e.g., frequent patterns, strong rules, approximate inclusion or functional dependencies) but also to some properties of these patterns in the selected database (e.g., their frequencies, the error for approximate dependencies). In that case, we will consider the so-called *extended theories*. For instance, when mining frequent itemsets or frequent and valid association rules [2], the user needs for the frequency of the specified patterns or rules. Indeed, during the needed post-processing phase, the user/analyst often uses various objective interestingness measures like the confidence [2], the conviction [23] or the J-mesure [72] that are computed efficiently provided that the frequency of each frequent itemset is available. Otherwise, it might be extremely expensive to look at the data again.

Designing *solvers* for more or less primitive constraints concerns the core of data mining algorithmic research. We must have solvers that can compute the (extended) theories and that have good properties in practice (e.g., scalability w.r.t. the size of the database or the size of the search space). A "generate and test" approach that would enumerate the sentences of $\mathcal{L}$ and then test the selection predicate q is generally impossible. A huge effort has concerned a clever use of the constraints occurring in q to have a tractable evaluation of useful inductive queries. This is the research area of *constraint-based data mining*. Most of the algorithmic research in pattern discovery tackles the design

of complete algorithms for computing (extended) theories given more or less specific conjunctions of primitive constraints. Typically, many researchers have considered the computation of frequent patterns, i.e., patterns that satisfy a minimal frequency constraint. An important paper on a generic algorithm for such a typical mining task is [55]. However, if the active use of the so-called *anti-monotonic constraints* (e.g., the minimal frequency) is now well-understood, the situation is far less clear for non anti-monotonic constraints [64,51,18].

A second major issue is the possibility to approximate the results of (extended) inductive queries. This approximation can concern a collection of patterns that is a superset or a subset of the desired collection. This is the typical case when the theories are computed from a sample of the data (see, e.g., [74]) or when a relaxed constraint is used. Another important case of approximation for extended theories is the exact computation of the underlying theory while the evaluation functions are only approximated. This has lead to an important research area, the computation of the so-called *condensed representations* [54], a domain in which we have been playing a major role since the study of frequent closed itemsets as an ϵ-adequate representation for frequency queries [12].

This paper is organized as follows. Section 2 introduces notations and definitions that are needed for discussing inductive queries that return itemsets. It contains an instance of the definition of a pattern domain. Section 3 identifies several important open problems. Section 4 provides elements of solution that are currently studied within the cInQ project. Section 5 is a short conclusion.

2 A Pattern Domain for Itemsets

The definition of a *pattern domain* is made of the definition of a language of patterns $\mathcal{L}$, evaluation functions that assign a semantics to each pattern in a given database $\mathbf{r}$, languages for primitive constraints that specify the desired patterns, and inductive query languages that provide a language for combining the primitive constraints.

We do not claim that this paper is an exhaustive description of the itemset pattern domain. Even though we selected representative examples of evaluation functions and primitive constraints, many others have been or might be defined and used.

2.1 Language of Patterns and Terminology

We introduce some notations that are used for defining the pattern domain of itemsets. In that context, we consider that:

- A so-called *transactional database* contains the data,
- Patterns are the so-called *itemsets* and one kind of descriptive rule that can be derived from them, i.e., the *association rules*.

Definition 1 (Transactional Databases). *Assume that* `Items` *is a finite set of symbols denoted by capital letters, e.g.,* $\texttt{Items} = \{\texttt{A}, \texttt{B}, \texttt{C}, \ldots\}$. *A* transaction

t is a subset of $\mathtt{Items}$*. A* transactional database $\mathbf{r}$ *is a finite and non empty multiset* $\mathbf{r} = \{t_1, t_2, \ldots, t_n\}$ *of transactions.*

Typical examples of transactional databases concern basket data (transactions are sets of products that are bought by customers), textual data (transactions are sets of keywords or descriptors that characterize documents), or gene expression data (transactions are sets of genes that are over-expressed in given biological conditions).

Definition 2 (Itemsets). *An* itemset *is a subset of* $\mathtt{Items}$*. The language of patterns for itemsets is* $\mathcal{L} = 2^{\mathtt{Items}}$*.*

We often use a string notation for sets, e.g., $\mathtt{AB}$ for $\{\mathtt{A}, \mathtt{B}\}$. Figure 1 provides an example of a transactional database and some information about itemsets within this database.

Association rules are not only a classical kind of pattern derived from itemsets [1,2] but are also used for some important definitions.

Definition 3 (Association Rules). *An association rule is denoted* $X \Rightarrow Y$ *where* $X \cap Y = \emptyset$ *and* $X \subseteq \mathtt{Items}$ *is the* body *of the rule and* $Y \subseteq \mathtt{Items}$ *is the* head *of the rule.*

Let us now define constraints on itemsets.

Definition 4 (Constraint). *If* $\mathcal{T}$ *denotes the set of all transactional databases and* $2^{\mathtt{Items}}$ *the set of all itemsets, an itemset constraint* $\mathcal{C}$ *is a predicate over* $2^{\mathtt{Items}} \times \mathcal{T}$*. An itemset* $S \in 2^{\mathtt{Items}}$ satisfies *a constraint* $\mathcal{C}$ *in the database* $\mathbf{r} \in \mathcal{T}$ *iff* $\mathcal{C}(S, \mathbf{r}) = true$*. When it is clear from the context, we write* $\mathcal{C}(S)$*. Given a subset* I *of* $\mathtt{Items}$*, we define* $\mathrm{SAT}_{\mathcal{C}}(I) = \{S \in I, S \text{ satisfies } \mathcal{C}\}$*.* $\mathrm{SAT}_{\mathcal{C}}$ *denotes* $\mathrm{SAT}_{\mathcal{C}}(2^{\mathtt{Items}})$*. The same definitions can be easily extended to rules.*

2.2 Evaluation Functions

Evaluation functions return information about the properties of a given pattern in a given database. Notice that using these evaluation functions can be considered as a useful task for the user/analyst. It corresponds to *hypothesis testing* when hypothesis can be expressed as itemsets or association rules, e.g., what are the transactions that support the H hypothesis? How many transactions support H? Do I have less than n counter-examples for hypothesis H?

Several evaluation functions are related to the "satisfiability" of a pattern in a given data set, i.e., deciding whether a pattern hold or not in a given database.

Definition 5 (Support for Itemsets and Association Rules). *A transaction* t supports *an itemset* X *if every item in* X *belongs to* t*, i.e.,* $S \subseteq t$*. It is then possible to define a boolean evaluation function* e_1 *such that* $e_1(X, \mathbf{r})$ *is true if all the transactions in* $\mathbf{r}$ *support* X *and false elsewhere. The same definition can be adapted for association rules:* $e_1(X \Rightarrow Y, \mathbf{r})$ *returns true iff when* $\mathbf{r}$ *supports* X*, it supports* Y *as well. The* support *(denoted* $\mathrm{support}(S, \mathbf{r})$*) of an itemset* S *is*

the multiset of all transactions of $\mathbf{r}$ *that supports* S *(e.g.,* $\text{support}(\emptyset) = \mathbf{r}$*). The* support *of a rule is defined as the support of the itemset* $X \cup Y$*. A transaction* t supports *a rule* $X \Rightarrow Y$ *if it supports* $X \cup Y$*.*

Definition 6 (Exceptions to Rules). *A transaction* t *is an exception for a rule* $X \Rightarrow Y$ *if it supports* X *and it does not support* Y*. It is then possible to define a new boolean evaluation function* $e_2(X \Rightarrow Y, \mathbf{r})$ *that returns true if none of the transactions* t *in* $\mathbf{r}$ *is an exception to the rule* $X \Rightarrow Y$*. A rule with no exception is called a* logical *rule.*

These evaluation functions that return sets of transactions or boolean values are useful when crossing over the patterns and the transactional data. Also, the size of the supporting set is often used.

Definition 7 (Frequency). *The* absolute frequency *of an itemset* S *in* $\mathbf{r}$ *is defined by* $\mathcal{F}_a(S, \mathbf{r}) = |\text{support}(S)|$ *where* $|.|$ *denote the cardinality of the multiset (each transaction is counted with its multiplicity). The relative frequency of* S *in* $\mathbf{r}$ *is* $\mathcal{F}(S, \mathbf{r}) = |\text{support}(S)|/|\text{support}(\emptyset)|$*. When there is no ambiguity from the context, parameter* $\mathbf{r}$ *is omitted and the frequency denotes the relative frequency (i.e., a number in [0,1]).*

Figure 1 provides an example of a transactional database and the supports and the frequencies of some itemsets.

$$T = \begin{array}{c|c} t_1 & \texttt{ABCD} \\ t_2 & \texttt{BC} \\ t_3 & \texttt{AC} \\ t_4 & \texttt{AC} \\ t_5 & \texttt{ABCD} \\ t_6 & \texttt{ABC} \end{array}$$

Itemset	Support	Frequency
A	$\{t_1, t_3, t_4, t_5, t_6\}$	0.83
B	$\{t_1, t_2, t_5, t_6\}$	0.67
AB	$\{t_1, t_5, t_6\}$	0.5
AC	$\{t_1, t_3, t_4, t_5, t_6\}$	0.83
CD	$\{t_1, t_5\}$	0.33
ACD	$\{t_1, t_5\}$	0.33

Fig. 1. Supports and frequencies of some itemsets in a transactional database

Other measures might be introduced for itemsets that, e.g., returns the degree of correlation between the attributes it contains. We must then provide evaluation functions that compute these measures.

It is straightforward to define the frequency evaluation function of an association rule $X \Rightarrow Y$ in $\mathbf{r}$ as $\mathcal{F}(X \Rightarrow Y, \mathbf{r}) = \mathcal{F}(X \cup Y, \mathbf{r})$ [1,2]. When mining association rules, we often use objective interestingness measures like confidence [2], conviction [23], J-mesure [72], etc. These can be considered as new evaluation functions. Most of these measures can be computed from the frequencies of rule components. For instance, the *confidence* of a rule $X \Rightarrow Y$ in $\mathbf{r}$ is $\text{conf}(X \Rightarrow Y) = \mathcal{F}(X \Rightarrow Y, \mathbf{r})/\mathcal{F}(Y, \mathbf{r})$. It gives the conditional probability that a transaction from $\mathbf{r}$ supports $X \cup Y$ when it supports X. The confidence of a logical rule is thus equal to 1.

We consider now several evaluation functions that have been less studied in the data mining context but have been proved quite useful in the last 3 years

(see, e.g., [65,12,75,15,16]). Notice however that these concepts have been used for a quite a long time in other contexts, e.g., in concept lattices.

Definition 8 (Closures of Itemsets). *The closure of an itemset S in $\mathbf{r}$ (denoted by $\texttt{closure}(S, \mathbf{r})$) is the maximal (for set inclusion) superset of S which has the same support as S. In other terms, the closure of S is the set of items that are common to all the transactions which support S.*

Notice that when the closure of an itemset X is a proper superset of X, say Y, it means that an association rule $X \Rightarrow Y \setminus X$ holds in $\mathbf{r}$ with confidence 1.

Example 1. In the database of Figure 1, let us compute $\texttt{closure}(\texttt{AB})$. Items A and B occur in transactions 1, 5 and 6. Item C is the only other item that is also present in these transactions, thus $\texttt{closure}(\texttt{AB}) = \texttt{ABC}$. Also, $\texttt{closure}(\texttt{A}) = \texttt{AC}$, $\texttt{closure}(\texttt{B}) = \texttt{BC}$, and $\texttt{closure}(\texttt{BC}) = \texttt{BC}$.

We now introduce an extension of this evaluation function [15,16].

Definition 9 (δ-closure). *Let δ be an integer and S an itemset. The δ-closure of S, $\texttt{closure}_\delta(S)$ is the maximal (w.r.t. the set inclusion) superset Y of S such that for every item $A \in Y - S$, $|\texttt{Support}(S \cup \{\texttt{A}\})|$ is at least $|\texttt{Support}(S)| - \delta$. In other terms, $\mathcal{F}_a(\texttt{closure}_\delta(S))$ has almost the same value than $\mathcal{F}_a(S)$ when δ is small w.r.t. the number of transactions.*

Example 2. In the database of Figure 1, $\texttt{closure}_2(\texttt{B}) = \texttt{BCD}$ while $\texttt{closure}_0(\texttt{B}) = \texttt{BC}$.

Notice that $\texttt{closure}_0 = \texttt{closure}$. Also, the δ-closure of a set X provides an association rule with high confidence between X and $\texttt{closure}_\delta(X) \setminus X$ when δ is a positive integer that is small w.r.t. the number of transactions.

It is of course possible to define many other evaluation functions. We gave representative examples of such functions and we now consider examples of primitive constraints that can be built from them.

2.3 Primitive Constraints

Many primitive constraints can be defined. We consider some examples that have been proved useful. These examples are representative of two important kinds of constraints: constraints based on evaluation functions and syntactic constraints. These later can be checked without any access to the data and are related to the well known machine learning concept of linguistic bias (see, e.g., [63]).

Let us consider primitive constraints based on frequency. First, we can enforce that a given pattern is frequent enough ($\mathcal{C}_{\text{minfreq}}(S)$) and then we specify that a given pattern has to be infrequent or not too frequent ($\mathcal{C}_{\text{maxfreq}}(S)$).

Definition 10 (Minimal Frequency). *Given an itemset S and a frequency threshold $\gamma \in [0, 1]$, $\mathcal{C}_{\text{minfreq}}(S) \equiv \mathcal{F}(S) \geq \gamma$. Itemsets that satisfy $\mathcal{C}_{\text{minfreq}}$ are said γ-*frequent *or frequent in $\mathbf{r}$. Indeed, this constraint can be defined also on association rules: $\mathcal{C}_{\text{minfreq}}(X \Rightarrow Y) \equiv \mathcal{F}(X \Rightarrow Y) \geq \gamma$.*

Definition 11 (Maximal Frequency). *Given an itemset S and a frequency threshold $\gamma \in [0,1]$, $\mathcal{C}_{\text{maxfreq}}(S) \equiv \mathcal{F}(S) \leq \gamma$. Indeed, this constraint can be defined also on association rules.*

Definition 12 (Minimal Confidence on Rules). *Given a rule $X \Rightarrow Y$ and a confidence threshold $\theta \in [0,1]$, $\mathcal{C}_{\text{minconf}}(X \Rightarrow Y) \equiv \text{conf}(S) \geq \theta$. Rules that satisfy $\mathcal{C}_{\text{minconf}}$ are called valid rules. A dual constraint for maximal confidence might be introduced as well.*

Example 3. Considering the database of Figure 1, if $\mathcal{C}_{\text{minfreq}}$ specifies that an itemset (or a rule) must be 0.6-frequent, then $\text{SAT}_{\mathcal{C}_{\text{minfreq}}} = \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{AC}, \texttt{BC}\}$. For rules, if the confidence threshold is 0.7, then the frequent and valid rules are $\text{SAT}_{\mathcal{C}_{\text{minfreq}} \wedge \mathcal{C}_{\text{minconf}}}(\emptyset \Rightarrow A, \emptyset \Rightarrow C, \emptyset \Rightarrow AC, A \Rightarrow C, C \Rightarrow A, B \Rightarrow C)$.

It is straightforward to generalize these definitions to all the other insterestingness measures that we can use for itemsets and rules. However, let us notice that not all the interestingness measures have bounded domain values. It motivates the introduction of the optimal constraints.

Definition 13 (Optimality). *Given an evaluation function $\mathcal{E}$ that returns an ordinal value, let us denote by $\mathcal{C}_{opt}(\mathcal{E}, \phi, n)$ the constraint that is satisfied if ϕ belongs to the n best patterns according to $\mathcal{E}$ values (the n patterns with the highest values).*

For instance, such a constraint can be used to specify that only the n most frequent patterns are desired (see [7,70,71] for other examples).

Another kind of primitive constraint concerns the *syntactical restrictions* that can be defined on one pattern. By syntactical, we mean constraints that can be checked without any access to the data and/or the background knowledge, just by looking at the pattern. A systematic study of syntactical constraints for itemsets and rules has been described in [64,51].

Definition 14 (Syntactic Constraints). *It is of the form $S \in \mathcal{L}_C$, where $\mathcal{L}_C \subseteq \mathcal{L} = 2^{\texttt{Items}}$. Various means can be used to specify $\mathcal{L}_C$, e.g., regular expressions.*

Some other interesting constraints can use additional information about the items, i.e., some background knowledge encoded in, e.g., relational tables. In [64], the concept of *aggregate constraint* is introduced.

Definition 15 (Aggregate Constraint). *It is of the form agg(S)θv, where agg is one of the aggregate functions min, max, sum, count, avg, and θ is one of the boolean operators $=, \neq, <, \leq, >, \geq$. It says the aggregate of the set of numeric values in S stands in relationship θ to v.*

Example 4. Consider the database of Figure 1, assume that $\mathcal{C}_{size}(S) \equiv |S| \leq 2$ (it is equivalent to $count(S) \leq 2$) and $\mathcal{C}_{miss}(S) \equiv \texttt{B} \notin S$, then $\text{SAT}_{\mathcal{C}_{size}} = \{\emptyset, \texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}, \texttt{AB}, \texttt{AC}, \texttt{AD}, \texttt{BC}, \texttt{BD}, \texttt{CD}\}$ and $\text{SAT}_{\mathcal{C}_{miss}} = \{\emptyset, \texttt{A}, \texttt{C}, \texttt{D}, \texttt{AC}, \texttt{AD}, \texttt{ACD}\}$.

The same kind of syntactical constraint can be expressed on association rules, including the possibility to express constraints on the body and/or the head of the rule.

We now consider primitive constraints based on closures.

Definition 16 (Closed Itemsets and Constraint $\mathcal{C}_{close}$). *A closed itemset is an itemset that is equal to its closure in* $\mathbf{r}$. *Let us assume that* $\mathcal{C}_{close}(S) \equiv \texttt{closure}(S) = S$. *In other terms, closed itemsets are maximal sets of items that are supported by a multiset of transactions.*

Example 5. In the database of Figure 1, the closed itemsets are `C`, `AC`, `BC`, `ABC`, and `ABCD`.

Free itemsets are sets of items that are not "strongly" correlated [15]. They have been designed as a useful intermediate representation for computing closed sets since the closed sets are the closures of the free sets.

Definition 17 (Free Itemsets and Constraint $\mathcal{C}_{free}$). *An itemset* S *is free if no logical rule holds between its items, i.e., it does not exist two distinct itemsets* X, Y *such that* $S = X \cup Y$, $Y \neq \emptyset$ *and* $X \Rightarrow Y$ *is a logical rule.*

Example 6. In the database of Figure 1, the free sets are $\emptyset$, `A`, `B`, `D`, and `AB`.

An alternative definition is that all the proper subsets of a free set S have a different frequency than S. Notice that free itemsets have been formalized independently as the co-called *key* patterns [5]. Furthermore, the concept of free itemset formalizes the concept of generator [65] in an extended framework since free itemsets are a special case of δ-free itemsets [15,16].

Definition 18 (δ-free Itemsets and Constraint $\mathcal{C}_{\delta-free}$). *Let* δ *be an integer and* S *an itemset, an itemset* S *is* δ*-free if no association rule with at most* δ *exceptions holds between its subsets.* δ*-free sets satisfy the constraint* $\mathcal{C}_{\delta-free}(S) \equiv (\forall S' \subset S) \Rightarrow S \not\subseteq \texttt{closure}_\delta(S')$.

Example 7. In the database of Figure 1, the 1-free sets are $\emptyset$, `A`, `B`, and `D`.

2.4 Example of Inductive Queries

Now, it is interesting to consider boolean combinations of primitive constraints. Notice that in this section, we consider neither the problem of query evaluation nor the availability of concrete query languages.

The Standard Association Rule Mining Task. Mining the frequent itemsets means the computation of $\text{SAT}_{\mathcal{C}_{\text{minfreq}}}$ for a given frequency threshold. The standard association rule mining problem introduced in [1] is to find all the association rules that verify the minimal frequency and minimal confidence constraints for some user-defined thresholds. In other terms, we are looking for each pattern ϕ (rules) such that $\mathcal{C}_{\text{minfreq}}(\phi) \wedge \mathcal{C}_{\text{minconf}}(\phi)$ is true. Filtering rules according to syntactical criteria can also be expressed by further constraints.

Example 8. Provided the dataset of Figure 1 and the constraints from Example 3 and 4, $\text{SAT}_{\mathcal{C}_{\text{minfreq}} \wedge \mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\mathtt{A}, \mathtt{C}, \mathtt{AC}\}$ is returned when the query specifies that the desired itemsets must be 0.6-frequent, with size less than 3 and without the attribute B. It is straightforward to consider queries on rules. Let us consider an example where the user/analyst wants all the frequent and valid association rules but also quite restricted rules with high confidence (but without any minimal frequency constraint). Such a query could be based, e.g., on the constraint $(\mathcal{C}_{\text{minfreq}}(\phi) \wedge \mathcal{C}_{\text{minconf}}(\phi)) \vee (\mathcal{C}_s(\phi) \wedge \mathcal{C}_{\text{minconf}}(\phi))$ where $\mathcal{C}_s(X \Rightarrow Y) \equiv |X| = |Y| = 1 \wedge Y = \mathtt{A}$.

Mining Discriminant Patterns. An interesting application of frequency constraints concerns the search for patterns that are frequent in one data set and infrequent in another one. This has been studied in [33] as the emerging pattern mining task. More recently, it has been studied within an inductive logic programming setting in [32] and applied to molecular fragment discovery.

Assume a transactional database for which one of the item defined a class value (e.g., item A is present when the transaction has the class value "interesting" and false when the transaction has the class value "irrelevant"). It is then possible to split the database $\mathbf{r}$ into two databases, the one of interesting transactions $\mathbf{r}_1$ and the one of irrelevant transactions (say $\mathbf{r}_2$). Now, a useful mining task concerns the computation of every itemset such that $\mathcal{C}_{\text{minfreq}}(S, \mathbf{r}_1) \wedge \mathcal{C}_{\text{maxfreq}}(S, \mathbf{r}_2)$. Indeed, these itemsets are supported by interesting transactions and not supported by irrelevant ones. Thresholds can be assigned thanks to a statistical analysis and such patterns can be used for predictive mining tasks.

Mining Association Rules with Negations. Let $\mathtt{Items}^+ = \{\mathtt{A}, \mathtt{B}, ...\}$ be a finite set of symbols called the positive items and a set $\mathtt{Items}^-$ of same cardinality as $\mathtt{Items}^+$ whose elements are denoted $\overline{\mathtt{A}}$, $\overline{\mathtt{B}}$, ... and called the negative items. Given a transaction database $\mathbf{r}$ over $\mathtt{Items}^+$, let us define a complemented transaction database over $\mathtt{Items} = \mathtt{Items}^+ \cup \mathtt{Items}^-$ as follows: for a given transaction $t \in \mathbf{r}$, we add to t negative items corresponding to positive items not present in t. Generalized itemsets are subsets of $\mathtt{Items}$ and can contain positive and negative items. In other terms, we want to have a symmetrical impact for the presence or the absence of items in transactions [14]. It leads to extremely dense transactional databases, i.e., extremely difficult extraction processes.

In [13], the authors studied the extraction of frequent itemsets ($\mathcal{C}_{\text{minfreq}}$) that do not involve only negative items ($\mathcal{C}_{alpp}$). $\mathcal{C}_{alpp}(S)$ is true when S involves at least p positive items. Also, this constraint has been relaxed into $\mathcal{C}_{alppoam1n} = \mathcal{C}_{alpp} \vee \mathcal{C}_{am1n}$ (at least p positive attributes or at most 1 negative attribute). On different real data sets, it has been possible to get interesting results when it was combined with condensed representations (see Section 4).

Mining Condensed Representation of Frequent Itemsets. Condensed representation is a general concept (see, e.g., [54]) that can be extremely useful for the concise representation of the collection of frequent itemsets and their frequencies. In Section 3, we define more precisely this approach. Let us notice at that stage that several algorithms exist to compute various condensed

representations of the frequent itemsets: CLOSE [65], CLOSET[69], CHARM [75], MIN-EX [12,15,16], or PASCAL [5]. These algorithms compute different condensed representations: the frequent closed itemsets (CLOSE, CLOSET, CHARM), the frequent free itemsets (MIN-EX, PASCAL), or the frequent δ-free itemsets for MIN-EX. From an abstract point of view, these algorithms are respectively looking for itemsets that satisfy $\mathcal{C}_{\text{minfreq}} \wedge \mathcal{C}_{close}$, $\mathcal{C}_{\text{minfreq}} \wedge \mathcal{C}_{free}$, and $\mathcal{C}_{\text{minfreq}} \wedge \mathcal{C}_{\delta-free}$. Furthermore, it can be interesting to provide association rules whose components satisfy some constraints based on closures. For instance, association rules that are based on free itemsets on their left-hand side ($\mathcal{C}_{free}(BODY)$) and their closures on the right-hand side ($\mathcal{C}_{close}(BODY \cup HEAD)$) are of a particular interest: they constitute a kind of cover for the whole collection of frequent and valid association rules [4,8]. Notice also that the use of classification rules based on δ-free sets ($\mathcal{C}_{\delta-free}$) for the body and a class value in the head has been studied in [17,28].

Postprocessing Queries. Post-processing queries can be understood as queries on materialized collections of itemsets or association rules: the user selects the itemsets or the rules that fulfill some new criteria (while these itemsets or rules have been mined, e.g., they are all frequent and valid). However, from the specification point of view, they are not different from data mining queries even though the evaluation does not need an extraction phase and can be performed on materialized collections of itemsets or rules.

One important post-processing use is to cross over the patterns and the data, e.g., when looking at transactions that are exceptions to some rules. For instance, given an association rule $A \Rightarrow B$, one wants all the transactions t from $\mathbf{r}$ (say the transactional database $\mathbf{r}_1$ for which $e_2(A \Rightarrow B, t)$ is true. Notice that $\mathbf{r} \setminus \mathbf{r}_1$ is the collection of exceptions to the rule. A rule mining query language like `MSQL` [43] offers a few built-in primitives for rule post-processing, including primitives that cross-over the rules and the transactions (see also [10] in this volume for examples of post-processing queries).

3 A Selection on Some Open Problems

We consider several important open problems that are related to itemset and rule queries.

3.1 Tractability of Frequent Itemset and Association Rule Mining

Computing the result of the classical association rule mining problem is generally done in two steps [2]: first the computation of all the frequent itemsets and their frequency and then the computation of every valid association rule that can be made from disjoint subsets of each frequent itemset. This second step is far less expensive than the first one because no access to the database is needed: only the collection of the frequent itemsets and their frequencies are needed. Furthermore, the frequent itemsets can be used for many other applications, far beyond the classical association rule mining task. Notice among others, clustering (see, e.g., [61]), classification (see, e.g., [53,28]), generalized rule mining (see, e.g., [54,14]).

Computing the frequent itemsets is an important data mining task that has been studied by many researchers since 1994. The famous APRIORI algorithm [2] has inspired many research and efficient implementations of APRIORI-like algorithms can be used provided that the collection of the frequent itemsets is not too large. In other terms, for the desired frequency threshold, the size of the maximal frequent itemsets must not be too long (around 15). Indeed, this kind of algorithm must count the frequencies of at least every frequent itemset and useful tasks, according to the user/analyst, become intractable as soon as the size of $SAT_{\mathcal{C}_{\text{minfreq}}}$ is too large for the chosen frequency threshold. It is the case of dense and correlated datasets and many real datasets fall in this category. In Section 4, we consider solutions thanks to the design of condensed representations for frequent itemsets.

The efficiency of the extraction of the answer to an itemset query relies on the possibility to use constraints during the itemset computation. A classical result is that effective safe pruning can be achieved when considering anti-monotonic constraints [55,64], e.g., the minimal frequency constraint. It relies on the fact that if an itemset violates an anti-monotonic constraint then all its supersets violate it as well and therefore this itemset and its supersets can be pruned and thus not considered for further evaluation.

Definition 19 (Anti-monotonicity). *An* anti-monotonic *constraint is a constraint* $\mathcal{C}$ *such that for all itemsets* S, S': $(S' \subseteq S \wedge \mathcal{C}(S)) \Rightarrow \mathcal{C}(S')$.

Example 9. Examples of anti-monotonic constraints are: $\mathcal{C}_{\text{minfreq}}(S)$, $\mathcal{C}(S) \equiv \mathtt{A} \notin S$, $\mathcal{C}(S) \equiv S \subseteq \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$, $\mathcal{C}(S) \equiv S \cap \{A, B, C\} = \emptyset$, $\mathcal{C}_{free}(S)$, $\mathcal{C}_{am1n}(S)$, $\mathcal{C}_{\delta-free}(S)$, $\mathcal{C}(S) \equiv S.price > 50$ and $\mathcal{C}(S) \equiv Sum(S.price) < 500$. The two last constraints mean respectively that the price of all items must be lower than fifty and that the sum of the prices of the items must be lower than five hundred.

Notice that the conjunction or disjunction of anti-monotonic constraints is anti-monotonic.

Even though the anti-monotonic constraints, when used actively, can drastically reduce the search space, it is not possible to ensure the tractability of an inductive query evaluation. In that case, the user/analyst has to use more selective constraints, e.g., a higher frequency threshold. Indeed, a side-effect can be that the extracted patterns become not enough interesting, e.g., they are so frequent that they correspond to trivial statements.

Furthermore, itemset queries do not involve only anti-monotonic constraints. For instance, $\mathcal{C}_{close}$ is not anti-monotonic. Sometimes, it is possible to post-process the collection of itemsets that satisfy the anti-monotonic part of the selection predicate to check the remaining constraints afterwards.

3.2 Tractability of Constraint-Based Itemset Mining

Pushing constraints is useful for anti-monotonic ones. Other constraints can be pushed like the *monotonic* constraints or the *succinct* constraints [64].

Definition 20 (Monotonicity). *A* monotonic *constraint is a constraint* $\mathcal{C}$ *such that for all itemsets* S, S': $(S \subseteq S' \wedge S$ *satisfies* $\mathcal{C}) \Rightarrow S'$ *satisfies* $\mathcal{C}$.

The negation of an anti-monotonic constraint is a monotonic constraint and the conjunction or disjunction of monotonic constraints is still monotonic.

Example 10. $C(S) \equiv \{A, B, C, D\} \subseteq S$, $\mathcal{C}_{alpp}(S)$, $C(S) \equiv Sum(S.price) > 100$ (the sum of the prices of items from S is greater than 100) and $C(S) \equiv S \cap \{A, B, C\} \neq \emptyset$ are examples of monotonic constraints.

Indeed, monotonic constraints can also be used to improve the efficiency of itemset extraction (optimization of the candidate generation phase that prevents to consider candidates that do not satisfy the monotonic constraint (see, e.g., [18]).

The succinctness property that has been introduced in [64] are syntactic constraints that can be put under the form of a conjunction of monotonic and anti-monotonic constraints. Clearly, it is possible to use such a property for the optimization of the constraint-based extraction (optimization of candidate generation and pruning).

Pushing non anti-monotonic constraints sometimes increases the computation times since it prevents effective pruning based on anti-monotonic constraints [73,18,34]. For instance, as described in [13], experiments have shown that it has been needed to relax the monotonic constraint $\mathcal{C}_{alpp}$ ("pushing" it gave rise to a lack of pruning) by $\mathcal{C}_{alppoam1n} = \mathcal{C}_{alpp} \vee \mathcal{C}_{am1n}$ where $\mathcal{C}_{am1n}$ is anti-monotonic. The identification of a good strategy for pushing constraints needs for an a priori knowledge of constraint selectivity. However, this is in general not available at extraction time. Designing adaptative strategies for pushing constraints during itemset mining is still an open problem. Notice however that some algorithms have been already proposed for specific strategies on itemset mining under conjunctions of constraints that are monotonic and anti-monotonic [64,18]. This has been explored further within the cInQ project (see Section 4).

Notice also that the constraints defined by a user on the desired association rules have to be transformed into suitable itemset constraints. So far, this has to be done by an ad-hoc processing and designing semi-automatic strategies for that goal is still an open problem.

3.3 Interactive Itemset Mining

From the user point of view, pattern discovery is an interactive and iterative process. The user defines a query by specifying various constraints on the patterns he/she wants. When a discovery process starts, it is difficult to figure out the collection of constraints that leads to an interesting result. The result of a data mining query is often unpredictable and the users have to produce sequences of queries until he/she gets an actionable collection of patterns. So, we have not only to optimize single inductive query evaluations but also the evaluation of sequences of queries. This has been studied for itemsets and association rules in, e.g., [37,3,36,62].

One classical challenge is the design of incremental algorithms for computing theories (e.g., itemsets that satisfy a complex constraint) or extended theories (e.g., itemsets and their frequencies) when the data changes. More generally, reusing previously computed theories to answer more efficiently to new inductive queries is important. It means that results about, equivalence and containment of inductive queries are needed. Furthermore, the concept of dominance has emerged [3]. In that case, only data scans are needed to update the value of the evaluation functions.

However, here again, a trade-off has to be found between the optimization of single queries by the best strategy for pushing its associated constraints and the optimization of the whole sequence. Indeed, the more we push the constraint and materialize only the constrained collection, the less it will be possible to reuse it for further evaluations [37,36]. In Section 4, we refer to recent advances in that area.

3.4 Concrete Query Languages

There is no dedicated query languages for itemsets but several proposal exist for association rule mining, e.g., (`MSQL` [43], `DMQL` [38], and `MINE RULE` [59]). Among them, the `MINE RULE` query language is one of the few proposals for which a formal operational semantics has been published [59]. Ideally, these query languages must support not only the selection of the mining context and its pre-processing (e.g., sampling, selection and grouping, discretization), the specification of a mining task (i.e., the expression of various constraints on the desired rules), and the post-processing of these rules (e.g., the support of subjective interestingness evaluation, redundancy elimination and grouping strategies, etc.).

A comparative study of the available concrete query languages is published in this volume [10]. It illustrates that we are still lacking from an "ideal" query language for supporting KDD processes based on association rules. From our perspective, we are still looking for a good set of primitives that might be supported by such languages. Furthermore, a language like `MINE RULE` enables to use various kinds of constraints on the desired rules (i.e., the relevant constraints on the itemsets are not explicit) and optimizing the evaluation of the mining queries (or sequences of queries) still need further research. This challenge is also considered within the cInQ project.

4 Elements of Solution

We now provide pointers to elements of solution that have been studied by the cInQ partners these last 18 months. It concerns each of the issues we have been discussing in Section 3. Even though the consortium has not studied only itemsets but also molecular fragments, sequential patterns and strings, the main results can be illustrated on itemsets.

Let us recall that, again, we do not claim that this section considers all the solutions studied so far. Many other projects and/or research groups are interested in the same open problems and study other solutions. This is the

typical case for depth-first algorithms like [39] which opens new possibilities for efficient constraint-based itemset computation [67].

Let us first formalize that inductive queries that return itemsets might also provide the results of the frequencies for further use.

Definition 21 (Itemset Query). *A itemset query is a pair* $(\mathcal{C}, \mathbf{r})$ *where* $\mathbf{r}$ *is a transactional database and* $\mathcal{C}$ *is an itemset constraint. The result of a query* $Q = (\mathcal{C}, \mathbf{r})$ *is defined as the set* $Res(Q) = \{(S, \mathcal{F}(S)) \mid S \in SAT_{\mathcal{C}}\}$.

There are two main approaches for the approximation of $Res(Q)$:

- The result is $Approx(Q) = \{(S, \mathcal{F}(S)) \mid S \in SAT_{\mathcal{C}'}\}$ where $\mathcal{C}' \neq \mathcal{C}$. In that case, $Approx(Q)$ and $Res(Q)$ are different. When $\mathcal{C}$ is more selective in $\mathbf{r}$ than $\mathcal{C}'$, we have $Approx(Q) \subseteq Res(Q)$. A post-processing on $Approx(Q)$ might be used to eliminate itemsets that do not verify $\mathcal{C}$. When $\mathcal{C}$ is less selective than $\mathcal{C}'$ then $Approx(Q)$ is said incomplete.
- The result is $Approx(Q) = \{(S, \mathcal{F}'(S)) \mid S \in SAT_{\mathcal{C}}\}$ where $\mathcal{F}'$ provides an approximation of the frequency of each itemset in $Approx(Q)$.

Indeed, it can be so that the two situations occur simultaneously. A typical case is the use of sampling on the database: one can sample the database ($\mathbf{r}' \subset \mathbf{r}$ is the mining context) and compute $Res(Q)$ not in $\mathbf{r}$ but in $\mathbf{r}'$. In that case, both the collection of the frequent itemsets and their frequencies are approximated. Notice however that clever strategies can be used to avoid, in practice, an incomplete answer [74].

A classical result is that it is possible to represent the collection of the frequent itemsets by its maximal elements, the so-called positive border in [55] or the S set in the machine learning terminology [60]. Also, it is possible to compute these maximal itemsets and their frequencies without computing every frequency of every frequent itemsets (see, e.g., [6]). This can be generalized to any anti-monotonic constraint: the collection of the most specific sentences $Approx(Q)$ (e.g., the maximal itemsets) is a compact representation of $Res(Q)$ from which (a) it is easy to derive the exact collections of patterns (every sentence that is more general belongs to the solution, e.g., every subset of the maximal frequent itemsets) but, (b) the evaluation functions (e.g., the frequency) are only approximated. Thus, the maximal itemsets can be considered as an example of an approximative condensed representation of the frequent itemsets.

First, we have been studying algorithms that compute itemsets under more general constraints, e.g., conjunctions of anti-monotonic and monotonic constraints. Next, we have designed other approximative condensed representations and exact ones as well.

4.1 Algorithms for Constraint-Based Mining

cInQ partners have studied the extraction of itemsets (and rather similar pattern domains like strings, sequences or molecular fragments) under a conjunction of monotonic and anti-monotonic constraints. Notice also that since disjunctions

of anti-monotonic (resp. monotonic) constraints are anti-monotonic (resp. monotonic), it enables to consider rather general forms of inductive queries.

[46] provides a generic algorithm that generalizes previous work for constraint-based itemset mining in a levelwise approach (e.g., [73,64]). The idea is that, given a conjunction of an anti-monotonic constraint and a monotonic constraint ($\mathcal{C}_{am} \wedge \mathcal{C}_m$), it is possible to start a levelwise search from the minimal (w.r.t. set inclusion) itemsets that satisfy $\mathcal{C}_m$ and completes this collection until the maximal itemsets that satisfy the $\mathcal{C}_{am}$ constraint are reached. Such a levelwise algorithm provides the complete collection $Res(Q)$ when Q can be expressed by means of a conjunction $\mathcal{C}_{am} \wedge \mathcal{C}_m$. [46] introduces strategies (e.g., for computing the minimal itemsets that satisfy $\mathcal{C}_m$ by using the duality between monotonic and anti-monotonic constraints). Details are available in [44].

Mining itemsets under $\mathcal{C}_{am} \wedge \mathcal{C}_m$ can also be considered as a special case of the general algorithm introduced in [30]. This paper considers queries that are boolean expressions over monotonic and anti-monotonic primitives on a single pattern variable ϕ. This is a quite general form of inductive query and it is shown that the solution space corresponds to the union of various version spaces [60,57,40,41]. Because each version space can be represented in a concise way using its border sets S and G, [30] shows that the solution space of a query can be represented using the border sets of several version spaces. When a query enforces a conjunction $\mathcal{C}_{am} \wedge \mathcal{C}_m$, [30] proposes to compute $S(\mathcal{C}_{am} \wedge \mathcal{C}_m)$ as $\{s \in S(\mathcal{C}_{am}) \mid \exists g \in G(\mathcal{C}_m) : g \subseteq s\}$ and dually for $G(\mathcal{C}_{am} \wedge \mathcal{C}_m)$. Thus, the borders for $\mathcal{C}_{am} \wedge \mathcal{C}_m$ can be computed from $S(\mathcal{C}_{am})$ and from $G(\mathcal{C}_m)$ as usual for the classical version space approach. Sets such as $S(\mathcal{C}_{am})$ can be computed using classical algorithms such as the levelwise algorithm [55] and the dual set $G(\mathcal{C}_m)$ can be computed using the dual algorithms [32]. These border sets are an approximative condensed representation of the solution. For the MolFea specific inductive database, it has been proved quite effective for molecular fragment finding [49,32,50,48].

Sequential pattern mining has been studied as well. Notice that molecular fragments can be considered as a special case of sequences or strings. [27] studies sequential pattern mining under a specific conjunction of constraint that ask for minimal frequency and similarity w.r.t. a reference pattern. In this work, the main contribution has been to relax the similarity constraint into an anti-monotonic one to improve pruning efficiency. It is an application of the framework for convertible constraints [68]. Also, logical sequence mining under constraints has been studied, in a restricted framework [56] (regular expressions on the sequence of predicate symbols and minimal frequency) and in a more general setting [52] (conjunction of anti-monotonic and monotonic constraints).

4.2 Condensed Representations for Frequent Itemsets

cInQ partners have studied the condensed approximations in two complementary directions: the use of border sets in a very general setting (i.e., version spaces) but also several condensed representations of the frequent itemsets.

- Border sets represent the maximally general and/or maximally specific solutions to an inductive query. They can be used to bound the set of all

solutions [30]. This can be used in many different pattern domains provided that the search space is structured by a specialization relation and that the solution space is a version space. They are useful in case only membership of the solution set is important.
– Closed sets [65,12], δ-free sets [15,16], and disjoint-free sets [25] are condensed representations that have been designed as ϵ-adequate representations w.r.t. frequency queries, i.e., representations from which the frequency of any itemset can be inferred or approximated within a bounded error.

The collection of the γ-frequent itemsets and their frequencies can be considered as an $\gamma/2$-adequate representation w.r.t. frequency queries [12]. It means that the error on the inference of a frequency for a given itemset is bounded by $\gamma/2$. Indeed, the frequency of an infrequent itemset can be set to $\gamma/2$ while the frequency of a frequent one is known exactly. Given a set $\mathcal{S}$ of pairs $(X, \mathcal{F}(X))$, e.g., the collection of all the frequent itemsets and their frequencies, we are interested in condensed representations of $\mathcal{S}$ that are subsets of $\mathcal{S}$ with two properties: (1) They are much smaller than $\mathcal{S}$ and faster to compute, and (2), the whole set $\mathcal{S}$ can be generated from the condensed representation with no access to the database, i.e., efficiently.

We have introduced in Section 2.3 the concepts of closed sets, free sets and δ-free sets. Disjoint-free itemsets are a generalization of free itemsets [25]. They are all condensed representations of the frequent itemsets that are exact representations (no loss of information w.r.t. the frequent itemsets and their frequencies), except for the δ-free itemsets (with $\delta \neq 0$) which is an approximative one. Let us now give the principle of regeneration from the frequent closed itemsets:

– Given an itemset S and the set of frequent closed itemsets,
  • If S is not included in a frequent closed itemset then S is not frequent.
  • Else S is frequent and $\mathcal{F}(S) = \text{Max}\{\mathcal{F}(X),\ S \subseteq X \wedge \mathcal{C}_{close}(X)\}$.

As a result, γ-frequent closed itemsets are like the γ-frequent itemsets a $\gamma/2$-adequate representation for frequency queries.

Example 11. In the database of Figure 1, if the frequency threshold is 0.2, every itemset is frequent and the frequent closed sets are `C`, `AC`, `BC`, `ABC`, and `ABCD`. $\mathcal{F}(\texttt{AB}) = \mathcal{F}(\texttt{ABC})$ since `ABC` is the smallest closed superset of `AB`.

The regeneration from δ-free itemsets is provided later. By construction, $|SAT_{\mathcal{C}_{close}}| \leq |SAT_{\mathcal{C}_{free}}|$ and $|SAT_{\mathcal{C}_{\delta-free}}| \leq |SAT_{\mathcal{C}_{free}}|$ when $\delta > 0$. Also, in practice, the size of these condensed representations are several orders of magnitude lower than the size of the frequent itemsets for dense data sets [24].

Several algorithms exist to compute various condensed representations of frequent itemsets [65,69,75,12,15,5,25]. These algorithms compute different condensed representations: the frequent closed itemsets (Close, Closet, Charm), the frequent free itemsets (Min-Ex, Pascal), the frequent δ-free itemsets (Min-Ex), or the disjoint-free itemsets (H/VlinEx). Tractable extractions from dense and highly-correlated data have become possible for frequency thresholds on which previous algorithms are intractable.

Representations based on δ-free itemsets are quite interesting when it is not possible to mine the closed sets or even the disjoint-free sets, i.e., when the computation is intractable given the user-defined frequency threshold. Indeed, algorithms like CLOSE [65] or PASCAL [5] or H/VLIN-EX [25] use special kinds of logical rules to prune candidate itemsets because their frequencies can be inferred from the frequencies of others. However, to be efficient, these algorithms need that such logical rules hold in the data.

Let us now consider the δ-free itemsets and how they can be used to answer frequency queries. The output of the MIN-EX algorithm [16] is formally given by the three following sets: $FF(\mathbf{r}, \gamma, \delta)$ is the set of the γ-frequent δ-free itemsets, $IF(\mathbf{r}, \gamma, \delta)$ is the set of the minimal (w.r.t. the set inclusion) infrequent δ-free itemsets (i.e., the infrequent δ-free itemsets whose all subsets are γ-frequent). $FN(\mathbf{r}, \gamma, \delta)$ is the set of the minimal γ-frequent non-δ-free itemsets (i.e., the γ-frequent non-δ-free itemsets whose all subsets are δ-free). The two pairs (FF, IF) and (FF, FN) are two condensed representations based on δ-free itemsets.

It is possible to compute an approximation of the frequency of an itemset using one of these two condensed representations:

- Let S be an itemset. If there exists $X \in IF(\mathbf{r}, \gamma, \delta)$ such that $X \subseteq S$ then S is infrequent. If $S \notin FF(\mathbf{r}, \gamma, \delta)$ and there does not exist $X \in FN(\mathbf{r}, \gamma, \delta)$ such that $X \subseteq S$ then S is infrequent. In these two cases, the frequency of S can be approximated by $\gamma/2$ Else, let F be the δ-free itemset such that: $\mathcal{F}(F) = \text{Min}\{\mathcal{F}(X),\ X \subseteq S \text{ and } X \text{ is } \delta\text{-free}\}$. Assuming that $n_S = |\text{support}(S)|$ and $n_F = |\text{support}(F)|$, then $n_F \geq n_S \geq n_F - \delta(|S| - |F|)$, or, dividing this by n, the number of rows in $\mathbf{r}$, $\mathcal{F}(F) \geq \mathcal{F}(S) \geq \mathcal{F}(F) - \frac{\delta}{n}(|S| - |F|)$.

It is thus possible to regenerate an approximation of the answer to a frequent itemset query from one of the condensed representation (FF, IF) or (FF, FN). Typical δ values range from zero to a few hundreds. With a database size of several tens of thousands of rows, the error made is below few percents [16]. If $\delta = 0$, then the two condensed representations enable to regenerate exactly the answer to a frequent itemset query.

This line of work has inspired other researchers. For instance, [26] proposed a new exact condensed representation of the frequent itemsets that generalizes the previous ones. It is, to the best of our knowledge, the most interesting exact representation identified so far. In [66], new approximative condensed representations are proposed that are built from the maximal frequent itemsets for various frequency values.

The condensed representations can be used also for constraint-based mining of itemsets and the optimization of sequence of queries. In [46,45], constraint-based mining under conjunctions of anti-monotonic and monotonic constraints is combined with condensed representations. Some technical problems have to be solved and it has lead to the concept of *contextual δ-free itemsets* w.r.t. a monotonic constraint [19]. The use of condensed representations is not limited to the optimization of single queries. [47] describes the use of a cache that contains free itemsets to optimize the evaluation of sequences of itemset queries.

Notice that other researchers also consider the optimization of sequences based on condensed representations like the free itemsets [35].

4.3 Optimizing Association Rule Mining Queries

`MINE RULE` [59] has been designed by researchers who belong to the cInQ consortium. This is one of the query languages dedicated to association rule mining [9,10]. New extensions to the `MINE RULE` operator have been studied [11,58]. Two important and challenging new notions include: *pattern views* and relations among inductive queries. Both of these notions have also been included (and were actually inspired on `MINE RULE`) in the logical inductive database theory [29]. Pattern views intensionally specify a set of patterns using an inductive query in `MINE RULE`. This is similar in spirit to a traditional relation view in a traditional database. The view relation is defined by a query and can be queried like any other relation later on. It is the task of the (inductive) database management to take care (using, e.g., query materialization or query transformation) that the right answers are generated to such views. Pattern views raise many new challenges to data mining. The other notion that is nicely elaborated in `MINE RULE` concerns the dominance and subsumption relation between consecutive inductive queries. [11] studies the properties that the sets of patterns generated by two `MINE RULE` queries present in interesting situations. For instance, given two similar queries that are identical apart from one or more clauses in which they differ for an attribute, the result-sets of the two queries exhibit an inclusion relationship when a functional dependency is present between the differing attributes. [11] studies also the equivalence properties that two `MINE RULE` queries present when they have two clauses with constraints on attributes that are functionally dependent. Finally, it studies the properties that the queries have when multiple keys of a relation are involved. All these notions, if elaborated in the context of the inductive databases, will help the system to speed-up the query answering procedures. Again these ideas have been carried over to the logical theory of inductive databases [29].

Partners of the consortium have been inspired by the `MINE RULE` query language to study information discovery from XML data by means of association rules [22,21].

4.4 Towards a Theory of Inductive Databases

The final goal of the cInQ project is to propose a theory of inductive databases. As a first valuable step, a logical and set-oriented theory of inductive databases has been proposed [29,30,31], where the key idea is that a database consists of sets of data sets and sets of pattern sets, Furthermore there is an inductive query language, where each query either generates a data or a pattern set. Queries generating patterns sets are − in their most general form − arbitrary boolean expression over monotonic and anti-monotonic primitives. This corresponds to a logical view of inductive databases because the queries are boolean expressions as well as a set oriented one because the answers to inductive queries are sets of patterns.

Issues concerned with the evaluation and optimization of such inductive queries based on the border set representations can be found in [30]. Furthermore, various other issues concerned with inductive pattern views and the memory organization of such logical inductive databases are explored in [29]. Finally, various formal properties of arbitrary boolean inductive queries (e.g., normal forms, minimal number of version spaces needed) have been studied [31].

Interestingly, these theoretical results have emerged from an abstraction of useful KDD processes, e.g., for molecular fragment discovery with the domain specific inductive database MolFea [32,50,49,48] or for association rule mining processes with, e.g., the `MINE RULE` operator.

5 Conclusions

We provided a presentation of the itemset pattern domain. Any progress on constraint-based mining for itemsets can influence the research on the multiple uses of frequent itemsets (feature construction, similarity measures and clustering, classification rule mining or bayesian network construction, etc). It means that, not only (more or less generalized) association rule mining in difficult contexts like dense data sets can become tractable but also many other data mining processes can benefit from this outcome.

We introduced most of the results obtained by the cInQ consortium after 18 months of work. A few concepts have emerged that are now studied in depth, e.g., approximative and exact condensed representations, relationships between inductive query solutions and versions spaces, strategies for the active use of constraints during inductive query evaluation, containment and dominance between mining queries.

A lot has yet to be done, e.g., towards the use of these concepts for predictive data mining tasks. Also, we have to study the robustness of these concepts in various application domains and thus different pattern domains. It is a key issue to identify a set of data mining primitives and thus figure out what could be a good query language for inductive databases. Indeed, the design of dedicated inductive databases, e.g., inductive databases for molecular fragment discovery, is an invaluable step. Not only it solves interesting applicative problems but also it gives the material for abstraction and thus the foundations of the inductive database framework.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings SIGMOD'93*, pages 207–216, Washington, USA, May 1993. ACM Press.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

3. E. Baralis and G. Psaila. Incremental refinement of mining queries. In *Proceedings DaWaK'99*, volume 1676 of *LNCS*, pages 173–182, Firenze, I, Sept. 1999. Springer-Verlag.
4. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proceedings CL 2000*, volume 1861 of *LNCS*, pages 972–986, London, UK, 2000. Springer-Verlag.
5. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, Dec. 2000.
6. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings SIGMOD'98*, pages 85–93, Seattle, USA, May 1998. ACM Press.
7. R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proceedings SIGKDD'99*, pages 145–154, San Diego, USA, Aug. 1999. ACM Press.
8. C. Becquet, S. Blachon, B. Jeudy, J.-F. Boulicaut, and O. Gandrillon. Strong association rule mining for large gene expression data analysis: a case study on human SAGE data. *Genome Biology*, 3(12), Dec. 2002.
9. M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. A comparison between query languages for the extraction of association rules. In *Proceedings DaWaK'02*, volume 2454 of *LNCS*, pages 1–10, Aix-en-Provence, F, Sept. 2002. Springer-Verlag.
10. M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. Query languages supporting descriptive rule mining: a comparative study. In P. L. Lanzi and R. Meo, editors, *Database Support for Data Mining Applications*, number 2682 in LNCS. Springer-Verlag, 2003. This volume.
11. M. Botta, R. Meo, and M.-L. Sapino. Incremental execution of the MINE RULE operator. Technical Report RT 66/2002, Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185, I-10149 Torino, Italy, May 2002.
12. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, Apr. 2000. Springer-Verlag.
13. J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Mining association rules with negations. Technical Report 2000-14, INSA Lyon, LISI, Batiment Blaise Pascal, F-69621 Villeurbanne, France, Nov. 2000.
14. J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Towards the tractable discovery of association rules with negations. In *Proceedings FQAS'00*, Advances in Soft Computing series, pages 425–434, Warsaw, PL, Oct. 2000. Springer-Verlag.
15. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proceedings PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, Sept. 2000. Springer-Verlag.
16. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal*, 7(1):5–22, 2003.
17. J.-F. Boulicaut and B. Crémilleux. Delta-strong classification rules for predicting collagen diseases. In *Proceedings of the ECML-PKDD'01 Discovery Challenge on Thrombosis Data*, pages 29–38, Freiburg, D, Sept. 2001. Available on line.
18. J.-F. Boulicaut and B. Jeudy. Using constraint for itemset mining: should we prune or not? In *Proceedings BDA'00*, pages 221–237, Blois, F, Oct. 2000.
19. J.-F. Boulicaut and B. Jeudy. Mining free-sets under constraints. In *Proceedings IDEAS'01*, pages 322–329, Grenoble, F, July 2001. IEEE Computer Society.
20. J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the inductive database framework. In *Proceedings DaWaK'99*, volume 1676 of *LNCS*, pages 293–302, Firenze, I, Sept. 1999. Springer-Verlag.

21. D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. L. Lanzi. Discovering interesting information in XML data with association rules. In *Proceedings SAC 2003 Data Mining track*, Melbourne, USA, Mar. 2003. ACM Press.
22. D. Braga, A. Campi, M. Klemettinen, and P. L. Lanzi. Mining association rules from XML data. In *Proceedings DaWaK'02*, volume 2454 of *LNCS*, pages 21–30, Aix-en-Provence, F, Sept. 2002. Springer-Verlag.
23. S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings SIGMOD'97*, pages 265–276, Tucson, USA, May 1997. ACM Press.
24. A. Bykowski. *Condensed representations of frequent sets: application to descriptive pattern discovery*. PhD thesis, Institut National des Sciences Appliquées de Lyon, LISI, F-69621 Villeurbanne cedex, France, Oct. 2002.
25. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proceedings PODS'01*, pages 267–273. ACM Press, May 2001.
26. T. Calders and B. Goethals. Mining all non derivable frequent itemsets. In *Proceedings PKDD'02*, volume 2431 of *LNAI*, pages 74–83, Helsinki, FIN, Aug. 2002. Springer-Verlag.
27. M. Capelle, C. Masson, and J.-F. Boulicaut. Mining frequent sequential patterns under a similarity constraint. In *Proceedings IDEAL'02*, volume 2412 of *LNCS*, pages 1–6, Manchester, UK, Aug. 2002. Springer-Verlag.
28. B. Crémilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *Proceedings ES 2002*, pages 33–46, Cambridge, UK, Dec. 2002. Springer-Verlag.
29. L. de Raedt. A logical view of inductive databases. Technical report, Institut fur Informatik, Albert-Ludwigs-Universitat, Georges-Kohler-Allee, Gebaude 079, D-79110 Freiburg, Germany, May 2002. 13 pages.
30. L. de Raedt. Query evaluation and optimization for inductive database using version spaces (extended abstract). In *Proceedings DTDM'02 co-located with EDBT'02*, pages 19–28, Praha, CZ, Mar. 2002. An extended version appears in this volume.
31. L. de Raedt, M. Jaeger, S. D. Lee, and H. Mannila. A theory of inductive query answering (extended abstract). In *Proceedings ICDM'02*, pages 123–130, Maebashi City, Japan, December 2002. IEEE Computer Press.
32. L. de Raedt and S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *Proceedings IJCAI'01*, pages 853–862, Seattle, USA, Aug. 2001. Morgan Kaufmann.
33. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings SIGKDD'99*, pages 43–52, San Diego, USA, Aug. 1999. ACM Press.
34. M. N. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings VLDB'99*, pages 223–234, Edinburgh, UK, September 1999. Morgan Kaufmann.
35. A. Giacommetti, D. Laurent, and C. T. Diop. Condensed representations for sets of mining queries. In *Proceedings KDID'02 co-located with ECML-PKDD'02*, Helinski, FIN, Aug. 2002. An extended version appears in this volume.
36. B. Goethals and J. V. den Bussche. On supporting interactive association rule mining. In *Proceedings DaWaK'00*, volume 1874 of *LNCS*, pages 307–316, London, UK, Sept. 2000. Springer-Verlag.
37. B. Goethals and J. van den Bussche. A priori versus a posteriori filtering of association rules. In *Proceedings SIGMOD Workshop DMKD'99*, Philadelphia, USA, May 1999.

38. J. Han and M. Kamber. *Data Mining: Concepts and techniques*. Morgan Kaufmann Publishers, San Francisco, USA, 2000. 533 pages.
39. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings ACM SIGMOD'00*, pages 1–12, Dallas, Texas, USA, May 2000. ACM Press.
40. H. Hirsh. Theoretical underpinnings of version spaces. In *Proceedings IJCAI'91*, pages 665–670, Sydney, Australia, Aug. 1991. Morgan Kaufmann.
41. H. Hirsh. Generalizing version spaces. *Machine Learning*, 17(1):5–46, 1994.
42. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, Nov. 1996.
43. T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.
44. B. Jeudy. *Extraction de motifs sous contraintes: application à l'évaluation de requêtes inductives*. PhD thesis, Institut National des Sciences Appliquées de Lyon, LISI, F-69621 Villeurbanne cedex, France, Dec. 2002. In French.
45. B. Jeudy and J.-F. Boulicaut. Constraint-based discovery and inductive queries: application to association rule mining. In *Proceedings ESF Exploratory Workshop on Pattern Detection and Discovery*, volume 2447 of *LNAI*, pages 110–124, London, UK, Sept. 2002. Springer-Verlag.
46. B. Jeudy and J.-F. Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis journal*, 6:341–357, 2002.
47. B. Jeudy and J.-F. Boulicaut. Using condensed representations for interactive association rule mining. In *Proceedings PKDD'02*, volume 2431 of *LNAI*, pages 225–236, Helsinki, FIN, Aug. 2002. Springer-Verlag.
48. S. Kramer. Demand-driven construction of structural features in ILP. In *Proceedings ILP'01*, volume 2157 of *LNCS*, pages 132–141, Strasbourg, F, Sept. 2001. Springer-Verlag.
49. S. Kramer and L. de Raedt. Feature construction with version spaces for biochemical applications. In *Proceedings ICML'01*, pages 258–265, William College, USA, July 2001. Morgan Kaufmann.
50. S. Kramer, L. de Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proceedings SIGKDD'01*, pages 136–143, San Francisco, USA, Aug. 2001. ACM Press.
51. L. V. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proceedings SIGMOD'99*, pages 157–168, Philadelphia, USA, 1999. ACM Press.
52. S. D. Lee and L. de Raedt. Constraint-based mining of first order sequences in SEQLOG. In *Proceedings KDID'02 co-located with ECML-PKDD'02*, Helsinki, FIN, Aug. 2002. An extended version appears in this volume.
53. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings KDD'98*, pages 80–86, New York, USA, 1998. AAAI Press.
54. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings KDD'96*, pages 189–194, Portland, USA, Aug. 1996. AAAI Press.
55. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
56. C. Masson and F. Jacquenet. Mining frequent logical sequences with SPIRIT-LoG. In *Proceedings ILP'02*, volume 2583 of *LNAI*, pages 166–182, Sydney, Australia, July 2002. Springer-Verlag.
57. C. Mellish. The description identification problem. *Artificial Intelligence*, 52(2):151–168, 1992.

58. R. Meo. Optimization of a language for data mining. In *Proceedings of the 18th Symposium on Applied Computing SAC 2003 Data Mining track*, Melbourne, USA, Mar. 2003. ACM Press. To appear.
59. R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.
60. T. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1980.
61. P. Moen. *Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining*. PhD thesis, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Jan. 2000.
62. B. Nag, P. M. Deshpande, and D. J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *Proceedings SIGKDD'99*, pages 244–253, San Diego, USA, Aug. 1999. ACM Press.
63. C. Nedellec, C. Rouveirol, H. Ade, and F. Bergadano. Declarative bias in inductive logic programming. In L. de Raedt, editor, *Advances in Logic Programming*, pages 82–103. IOS Press, 1996.
64. R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings SIGMOD'98*, pages 13–24, Seattle, USA, 1998. ACM Press.
65. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, Jan. 1999.
66. J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. In *Proceedings ICDM'02*, pages 378–385, Maebashi City, JP, Dec. 2002. IEEE Computer Press.
67. J. Pei and J. Han. Constrained frequent pattern mining:a pattern-growth view. *SIGKDD Explorations*, 4(1):31–39, June 2002.
68. J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings ICDE'01*, pages 433–442, Heidelberg, D, Apr. 2001. IEEE Computer Press.
69. J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings SIGMOD Workshop DMKD'00*, Dallas, USA, May 2000.
70. T. Scheffer. Finding association rules that trade support optimally against confidence. In *Proceedings PKDD'01*, volume 2168 of *LNCS*, pages 424–435, Freiburg, D, Sept. 2001. Springer-Verlag.
71. J. Sese and S. Morishita. Answering the most correlated N association rules efficiently. In *Proceedings PKDD'02*, volume 2431 of *LNAI*, pages 410–422, Helsinki, FIN, Aug. 2002. Springer-Verlag.
72. P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, Aug. 1992.
73. R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings KDD'97*, pages 67–73, Newport Beach, USA, 1997. AAAI Press.
74. H. Toivonen. Sampling large databases for association rules. In *Proceedings VLDB'96*, pages 134–145, Mumbay, India, Sept. 1996. Morgan Kaufmann.
75. M. J. Zaki. Generating non-redundant association rules. In *Proceedings SIGKDD'00*, pages 34–43, Boston, USA, Aug. 2000. ACM Press.