Onto4AR: a framework for mining association rules

Cláudia Antunes[1]

[1] Instituto Superior Técnico / Technical University of Lisbon
Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal
claudia.antunes@dei.ist.utl.pt

Abstract. Despite the efforts made on last decades to center the process of knowledge discovery on the user, the balance between the discovery of unknown and interesting patterns is far from being reached. The discovery of association rules is a paradigmatic case, where this balance is quite difficult to establish. In this paper, we propose a new framework for pattern mining – the *Onto4AR*. This framework is centered on the use of ontologies, for the representation and introduction of domain knowledge into the mining process. By defining constraints based on an ontology, the framework provides a mining environment independent of the problem domain. With this simplification on the definition and use of constraints, the framework contributes to reduce the gap between discovered rules and user expectations.

Keywords: Association rules, Background knowledge, Ontologies, Constraints.

1 Introduction

In the information society, where the communication is instantaneous and the access to registered data is generalized, one of the major difficulties, faced by organizations, is related to the management of data abundance and its use on the decision support. Among the available tools to manage those large amounts of unstructured data, are *association rules*. The problem of its discovery was first introduced in 1993 [1], for finding sets of items purchased in the same transaction a significant number of times. Since then, association rules have been applied to a variety of domains, from medicine to marketing, with a fair success. Indeed, and despite the improvements on the algorithms for this task, the massive number of discovered rules turns their manual analysis impossible. On the other hand, the discovery of obvious rules diminishes the interest of users for this technique.

In this manner, a decade later, the discovery of association rules continues to present several challenges. On one hand, it is important to choose the interesting patterns to present to the user, filtering obvious rules; and on the other hand, it is mandatory that the user can control the mining process, in the sense that he should be able to incorporate his expectations and domain knowledge in the discovery process. Despite both issues have deserved considerable attention, namely in the definition of interestingness measures and languages to express constraints, their usage has not been exported to real situations, as desirable.

In this paper, we propose a new framework for discovering association rules – the *Onto4AR framework*. This framework provides an environment for specifying constraints that enables the user to control the mining process. But instead of using a specific language to introduce the constraints, it uses an ontology to represent existing background knowledge. Additionally, the user can choose the kind of constraint to be applied, both reducing the number of discovered patterns and their scope. The constraints can be chosen among a set of pre-defined ones based on the relations among concepts expressed in the ontology. In this manner, the framework provides a universal environment for mining association rules, since it is able to represent background knowledge in any domain, and supplies a set of parameterized constraints, defined independently of the problem context. Moreover, the framework allows for the definition of any new kind of constraints. In particular it is possible to introduce other content constraints, given that they are defined over the structure of ontologies.

The rest of the paper is organized as follows: next (in section 2), we overview a few interesting approaches that use background knowledge in the mining process, followed by a general description of ontologies (in section 3). In section 4, we describe the entire framework, presenting its main concepts and the adaptation of pattern mining algorithms to use proposed constraints. The section ends with a categorization of constraints that can be defined in the framework. The paper ends with the discussion of the benefits of the described approach and some guidelines for future work.

2 MINING RULES WITH CONSTRAINTS

The problem of mining association rules is defined as the discovery of "all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence respectively". With an *association rule* corresponding to an implication of the form $A \Rightarrow B$, where A and B are propositions (sets of pairs attribute/value, most of the times named *items*), and expresses that when A occurs, B also occurs with a certain probability. This probability is known as the rule *confidence* and is given by the conditional probability $P(B|A)$. The *support* of the same rule is given by the number of transactions that include A and B, simultaneously. In basket analysis, one of the major applications of association analysis, A and B correspond to itemsets, and the rule $A \Rightarrow B$ means that if A is transacted, then B will also be transacted at the same time, with a certain probability. The problem includes two separate phases: the discovery of frequent patterns and the generation of rules from those patterns. Since the second step demands the generation of all possible combinations among the items that constitute each pattern, research has been focused on the discovery of frequent patterns.

Since its definition, several algorithms have been proposed that efficiently deal with this problem, but most of them did not give any contribution to reduce the number of discovered rules. For example, most recent algorithms that find closed patterns (see for example CLOSET+ [15]), retrieve exactly the same patterns as traditional algorithms, like Apriori [1], despite they generate considerable less

patterns than those algorithms. In fact, the most effective technique to reduce the number of discovered patterns is the use of constraints. As pointed by Bayardo, constraints play a critical role in solving the trade-offs of the generality of data mining algorithms, by focusing "the algorithm on regions of the trade-off curves (or space) known (or believed) to be most promising" [5]. By using constraints, the user assumes the responsibility of choosing which of those aspects are most important for the current task.

In the problem of mining association rules, we can distinguish between two main categories of constraints: interestingness measures and content constraints. *Interestingness measures* rank the discovered patterns or rules, by quantifying the usefulness and utility of them, discarding those with an evaluation less than a user-specified threshold. Examples of such measures are *confidence*, *lift* or the *minimum improvement* [4]. With these constraints, it is possible to both improve the performance of the algorithms, by pruning uninteresting patterns, and reduce the number of discovered patterns. The other category of constraints, *content constraints*, can be seen as filters over the content of the discovered patterns, instead of its relevance. While interestingness measures are quantitative metrics, content constraints are predicates on the powerset of the set of items [14]. In some sense, they capture application semantics and introduce it into the mining process. An example of such constraints is the *item constraint*, as proposed by Srikant *et al* [16]. Despite the advantages coupled to the use of content constraints, by allowing the incorporation of domain knowledge, they have been applied seldom.

Despite the difficulties in applying existing background knowledge in real situations, in the last decade, there were a few proposals that contribute to develop this area. The first contribution was made by Srikant et al [16], which besides proposing item constraints, introduced the use of taxonomies to discover generalized patterns. A taxonomy is a *is-a* hierarchy defined over the domain in analysis and is often available as the unique background knowledge. By making use of taxonomies and specifying the items of interest, it is possible to focus the process on the user, either by transferring the control to him or by introducing part of his background knowledge in the mining process. With the same goal, but in the context of sequential pattern mining, Garofalakis [8] and Antunes [2] proposed the use of regular and context-free languages to constrain the discovery. Note that in areas, like sequential pattern mining, where constraints can be specified with such restrictive power, the risk of turning the mining process in a simple hypothesis-test becomes a reality. In order to solve this antagonism, more recently, the use of constraint relaxations was proposed [3].

Another important research way in the specification of constraints is the use of logic-based languages. Usually, these languages combine inductive and deductive perspectives of data mining and provide the means to introduce constraints in the mining process. The work by Bonchi [6] gives a good overview of such languages. These languages can also be used as a filter the number of discovered patterns, in a post-processing step, like the *mine rule* operator [7]. Again with post-processing in mind, Vanzin [17] proposed the use of Ontologies to manage the discovered patterns. In this work, the user can navigate over the patterns, following the knowledge represented in a domain ontology. With a wider range, some other works propose the use of ontologies for guiding the knowledge discovery process (KDD). The work by

Češpivová et al. [8] shows that ontologies can be used in all KDD phases, from business and data understanding to modeling, evaluation and deployment phases. Another example is the MiningMart system [10], which provides the tools for pre-processing data in order to facilitate the evaluation and usage of the mining results.

Next, we describe succinctly the core concepts in ontologies, and exemplify its use in the domain of basket analysis.

3. Ontologies

One of the great problems related to the use of background knowledge is the difficulty to acquire it complete and effectively, since, business agents usually have some difficulties on expressing their knowledge. Fortunately, the recent developments in the area of knowledge management make possible the representation of background knowledge as Ontologies.

An *ontology* is an explicit specification of a conceptualization, which means that is a specification of an abstract, simplified view of the domain. Formally, and in accordance with [12], an ontology is a 5-tuple $\mathcal{O}:=\{C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O\}$, where C is a set of concepts, who represent the entities in the ontology domain; $\mathcal{R}$ is a set of relations defined among concepts; $\mathcal{H}^C$ is a taxonomy or concept-hierarchy, which defines the is-a relations among concepts ($\mathcal{H}^C(C_1, C_2)$ means that C_1 is a sub-concept of C_2, or in other words C_2 is a parent of C_1); *rel* is a function, rel: $\mathcal{R} \rightarrow C \times C$, that specifies the relations on $\mathcal{R}$ (if $R \in \mathcal{R}$, $rel(R)=(C_1, C_2)$ is also written as $R(C_1, C_2)$, and means that C_1 is related to C_2, but the inverse is not necessarily true); and finally, $\mathcal{A}^O$ is a set of axioms, usually expressed in a logical language, that describe constraints on the ontology, expliciting implicit facts.

In the counterpart of ontologies are knowledge bases, which specify existing instantiations for a particular ontology. Formally, a *knowledge base* is a 4-tuple $\mathcal{KB}:=\{\mathcal{O}, \mathcal{I}, inst, instr\}$, where $\mathcal{O}$ is a ontology as defined above; $\mathcal{I}$ a set of instances; *inst* a function from C to $2^{\mathcal{I}}$ called *concept instantiation* (with C, the set of concepts in $\mathcal{O}$) and *instr* the relation instantiation function defined from $\mathcal{R}$ to $2^{\mathcal{I} \times \mathcal{I}}$. (It is usual to designate the concept instantiated by an instance, as its *class*). In this manner, while ontologies try "to capture the conceptual structures of a domain of interest" [12], defining its elements, and describing the relations among them; a knowledge base defines a set of elements, which can be interpreted and understand with respect to an ontology.

With the developments of the ontology research and the semantic web, several ontology specification languages have been proposed. See the work by Corcho et al [8] for an overview and comparison of such languages.

3.1 An Example

Fig. 1 illustrates a simple knowledge base $\mathcal{KB}$ constituted by a simple ontology $\mathcal{O}$, which main concepts are alimentary products and meals. The knowledge base is represented in UML, where *concepts* are represented by ellipses, *is-a relations* as

non-named arrows, and other *relations* by named stronger arrows. Double-sided arrows represent two inverse relations (just for draw simplification). Instances are represented by shadowed ellipses and instantiations by dotted arrows.

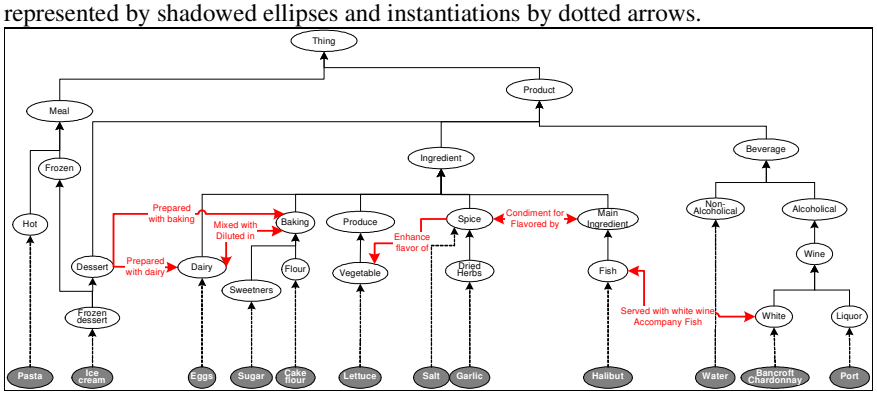

Fig. 1 Example of a knowledge base for alimentary products and meals

If we use the notation presented above, we can define the ontology as $\mathcal{O}$:={C, $\mathcal{R}$, $\mathcal{H}^C$, *rel*, $\mathcal{A}^O$}. With C:={*Thing, Product, Beverage, Non-Alcoholical, Alcoholical, Wine, White wine, Liquor, Ingredients, Dairy, Baking, Sweeteners, Flour, Main ingrediants, Fish, Produce, Vegetable, Spice, Dried herbs, Dessert, Frozen dessert, Meal, Frozen, Hot*}, with *Thing* the concept, from whom all other concepts descend. The set of relations $\mathcal{R}$:={*Condiment for, Flavored by, Enhance flavor of, Mixed with, Diluted in, Prepared with baking, Served with white wine, Accompany fish*}. The taxonomy $\mathcal{H}^C$:={(*Meal, Thing*), (*Frozen, Meal*), (*Hot, Meal*), (*Product, Thing*), (*Beverage, Product*), (*Non-Alcoholical, Product*), (*Alcoholical, Product*), (*Wine, Alcoholical*), (*White wine, Wine*), (*Ingredient, Product*), (*Dairy, Ingredient*), (*Baking, Ingredient*), (*Sweeteners, Baking*), (*Flour, Baking*), (*Ingredient, Product*), (*Fish, ingredient*), (*Produce, Ingredient*), (*Vegetable, Produce*), (*Spice, Ingredient*), (*Dried herbs, Spice*), (*Dessert, Product*), (*Frozen dessert, Dessert*), (*Frozen dessert, Frozen*)}. Finally, since there is no axiom, *rel* is defined by {*Condiment for (Spice, Main Ingredient), Flavored by (Vegetables, Spice), Enhance flavor of (Main Ingredient, Spice), Prepared with baking (Prepared dessert, Baking), Served with white wine (Fish dish, White wine), Accompany fish (White wine, Fish)*}.

The knowledge base $\mathcal{KB}$:={$\mathcal{O}$, $\mathcal{I}$, *inst*, *instr*}, with $\mathcal{O}$ defined as above, $\mathcal{I}$:={*Sugar, Eggs, Cake flour, Port, Bancroft Chardonnay, Water, Lettuce, Halibut, Salt, Garlic*} and *inst*:={*(Sugar, Baking), (Eggs, Dairy), (Cake flour, Flour), (Port, Liquor), (Bancroft Chardonnay, White wine), (Water, Non-alcoholical (Lettuce, Vegetable), (Halibut, Fish), (Salt, Spice), (Garlic, Dried herbs), (Pasta, Hot), (Ice cream, Frozen dessert)*}. Note that there is no relation instantiation.

From these, and above all, from the picture on Fig. 1, it is easy to understand that the ontology has two main concepts: *Product* and *Meal*. Moreover, among products *Ingredient*, *Dish* and *Beverage* are distinguishable. Each of these concepts has some sub-concepts, with particular emphasis to *Frozen dessert*, which derives from two distinct concepts: *Dessert* and *Frozen*.

Most of the instances in the corresponding knowledge base do not correspond to real products (with brand and model) due to the necessity of simplifying the picture.

4. The *Onto4AR* framework

Despite the developments on the creation of ontologies and knowledge representation, the large majority of mining projects does not make use of these well-formed conceptual models and despise most of the existing knowledge. On the other side, the improvements on the algorithms for mining association rules, least has contributed to incorporate background knowledge on the discovery process.

The balance between discovering unknown information and manageable quantities of patterns is far from being reached. But, as been shown, the incorporation of constraints (both in the form of interestingness measures and content constraints) is one of the few effective approaches to achieve that goal.

The *Onto4AR* framework, proposed in this paper, is a significant step in that pathway, since it defines a formal environment for incorporating background knowledge into the pattern mining process, that is independent on the problem domain, or the nature of the data. The framework is centered in a precisely definition of constraint and the problem of constrained pattern mining, based on the notions of ontology and knowledge base, presented before.

Next, the problem of mining association rules in the presence of background knowledge is defined.

4.1 Problem Statement

Consider a set of items $\mathcal{I}$, and an *itemset* $\mathcal{A}$ as a subset of $\mathcal{I}$ ($\mathcal{A} \subseteq \mathcal{I}$), and a *transaction* as an itemset that occurred in a particular instant of time. Additionally, consider that the $\mathcal{I}$ set corresponds to the set of instances of a knowledge base ($\mathcal{KB}$:={$\mathcal{O}$, $\mathcal{I}$, *inst*, *instr*}), that references an ontology $\mathcal{O}$. Suppose that *inst* is defined, which means that the concept implemented by each instance is known. The problem of mining association rules in the context of the *Onto4AR* framework can now be stated as:

Given a set of instances I from a specific knowledge base $\mathcal{KB}$={$\mathcal{O}$, $\mathcal{I}$, *inst*, *instr*}, *and a set of transactions D, find all rules of the form* $\mathcal{A} \Rightarrow \mathcal{B}$, *where* $\mathcal{A}$ *and* $\mathcal{B}$ *are disjoint itemsets of* $\mathcal{I}$ *that may occur in the transactions of* $\mathcal{D}$, *and the itemsets* $\mathcal{A}$ *and* $\mathcal{B}$ *satisfy a set of constraints* $C_{\mathcal{O}}$, *defined over the ontology* $\mathcal{O}$.

Note that this definition only differs from the usual one, because both support and confidence are constraints, included in the set of constraints $C_{\mathcal{O}}$. Also note that with this statement, there is no imposition on the relation between $\mathcal{A}$ and $\mathcal{B}$ (beside that they do not contain repeated items and all of them belong to the set of instances), neither on the number of times that they occur together or the confidence of the rule. All depends on the constraints imposed by $C_{\mathcal{O}}$. As in the rest of literature (see [13]), in the context of the *Onto4AR* framework, a *constraint* is a predicate on the powerset of the set of items $\mathcal{I}$, which means, that it is a function c: $2^{\mathcal{I}} \rightarrow$ {*true*, *false*}. An itemset S is said to satisfy c, if and only if, $c(S)$ is *true*. In this manner, a constraint imposes some condition over the elements of the itemset, or the relations among them. These predicates can establish either qualitative or quantitative conditions. Indeed, at a first glance, two different categories of constraints can be distinguished: *interestingness constraints* (also known as *interestingness measures*) and *content constraints*.

4.2. Interestingness Measures

Interestingness measures are constraints that impose quantitative conditions over the set of items in the rule, like the number of times that the set of items are transacted together, or the novelty introduced by the discovery of the rule. Formally, an interestingness measure is a composed function $f = f_\theta \circ g$ $[f(x)=g(f_\theta(x))]$, with $g: 2^I \rightarrow \mathbb{R}$ and $f_\theta: \mathbb{R} \rightarrow \{true, false\}$, defined by the comparison of its argument with θ, a threshold value.

A special interestingness measure is the *existential constraint*, usually known as *support*. For the rule A⇨B, it is simply defined by substituting *g* with the function *frequency and* f_θ with the function $>=_\theta$. Where the function *frequency* counts the number of transactions in the dataset that contain both the items of A and B, and the function $>=_\theta$ that returns *true* if and only if its argument is greater than or equal to θ, and *false* otherwise. Another usual interestingness measure is *confidence*, which in the *Onto4AR* framework can be defined by instantiating the *g* function by the conditional probability and the function f_θ again by $>=_\theta$. With the definition of these two interestingness measures, the problem of mining association rules as initially proposed [1] can be addressed in the *Onto4AR* framework, without any difficulty, since the set of items and the set of instances are the same things.

4.3. Content Constraints

Although interestingness constraints play a fundamental role on pattern mining, they only capture the knowledge about some quantity that is significant for the specific business. As such, they are not able to represent any other knowledge about the business domain. In order to go beyond this difficulty, content constraints introduce the ability of imposing that items in the rule have some specific characteristics. These characteristics can be selected among the ones represented in the domain ontology for the mining problem.

Formally, a *content constraint* can be defined as a predicate $c_{\mathcal{O}}: 2^I \rightarrow \{true, false\}$ that impose some qualitative condition on the items of its argument, expressed based in the domain knowledge represented in the ontology $\mathcal{O}$. For the rest of this section, consider the knowledge base $\mathcal{KB}:=\{\mathcal{O}, \mathcal{I}, inst, instr\}$, with $\mathcal{O}$ the ontology defined by the tuple $\{\mathcal{C}, \mathcal{R}, \mathcal{H}^c, rel, \mathcal{A}^o\}$.

The first kind of content constraint that can be defined in this framework is the item constraint. As defined by Srikant [16], an item constraint is a "Boolean expression over the presence or absence of items in the rule". And in order to be able to use abstract concepts to define the scope of item constraints in the new context, some additional notions are needed: parent, child, descendant and ancestor, like in that work [16]. In the *Onto4AR* context, *child* is a predicate defined over the concepts in the ontology $\mathcal{O}$, and *child* (x, y) is *true* if and only if $\mathcal{H}^c(x, y)$ is defined in the ontology. *Parent* is the inverse of *child*, and then *parent* (x, y) is *true* if and only if *child* (y, x) is *true*. The predicate *descendant* can be defined in a recursive way:

$$descendant(x,y) \Leftrightarrow child(x,y) \vee [\exists z \in C: child(x,z) \wedge descendant(z,y)],$$

which means that either x is child of y or exists a third concept z that is parent of x, and is descendant of y, simultaneously. With these definitions, it is now possible to define item constraints in a generic form. In the new context, an *item constraint* is a content constraint c_S, implemented by the predicate $\in_S$, and then it only returns *true* if some items of its argument belong to the user-defined subset S of I. Let an itemset $X \in 2^I$, it is said that

$$X \text{ satisfies an item constraint } c_S \Leftrightarrow \exists x \in X: x \in S$$

Note that it does not deal with the absence of some items, and the specification of S has to be entered manually by the user, as it was with the specification of the Boolean expression. This specification can be done either by enumerating all accepted instances, or by describing its elements properties. For example, using the ontology described on section 3, the item constraint defined over the set $\{x \in I: inst^{-1}(x) = Liquor\}$ will only accept patterns that include some liquors (with $inst^{-1}$ the inverse function of *inst*). Additionally, consider the notion of n^{th} *cousin*. An item is a n^{th} *cousin* of another item, if they are connected by a n degree collateral relation. More specifically,

$$\forall x, y \in I; n \in N : n^{th}cousin(x, y, n) = true \Leftrightarrow \begin{cases} descendant\ (x, parent(y))\ ,if\ \ n = 0 \\ n^{th}cousin(x,\ parent(y),\ n-1),\ otherwise \end{cases}$$

For example, x is a 0^{th} cousin of y if it descends from y's parent (x is either his "brother" or his "nephew"); x is a 1^{st} cousin of y, if it descends from y's grandparent (one degree of collaterality); x is a 2^{nd} cousin of y, if it descends from y's great-grandparent (two degrees of collaterality), and so on.

We can say that an item X is *at least an* n^{th}*-cousin* of another item Y if there is some $m \leq n$ such X is an m^{th}-cousin of Y. With these definitions, it is possible to define a new category of constraints – named *family constraints*. As expected, these constraints are implemented by the n^{th}*-cousin* and *at least* n^{th}*-cousin* predicates. From this category of constraints, it is possible to identify the *same-family*$_n$ constraint, which accepts an itemset if all their items share a common ancestor and all of them are at least n^{th} cousins to each other. This means that

$$X \text{ satisfies same-family}_n \Leftrightarrow \forall x,y \in X\ \exists m \in \mathbb{N}: x \neq y \wedge m \leq n \wedge n^{th}cousin(x,y,m)$$

Again, using the ontology described on section 3, the itemset {*Cake flour*, *Sugar*} satisfies the same-family constraint with n=0, since they share the same parent, and {*Port*, *Water*} satisfies the same-family constraint with n=2, since *Water* descends from the great-grandparent of *Port* (*Beverage*). But {*Port*, *Salt*} does not satisfy the constraint same-family with n=2, only for n=3. Note that every itemset satisfies the same-family constraint for unlimited n, since all concepts descend from the concept *Thing*. Similarly, the *close-family*$_n$ constraint accepts an itemset if every item in the set satisfies the *at least an* n^{th}*-cousin* predicate with another item in the set.

$$X \text{ satisfies close-family}_n \Leftrightarrow \forall x \in X\ \exists y \in X\ \exists m \in \mathbb{N}: x \neq y \wedge m \leq n \wedge n^{th}cousin(x,y,m)$$

In order to have an itemset that satisfies the close-family constraint and does not satisfy the same-family constraint, the itemset has to contain some instantiation of a concept that derives from more than one concept, this is, it has to have multiple parents. In previous ontology, only itemsets that contain instantiations of *Frozen dessert*, can satisfy the close-family constraint. For example the itemset {*Salt, Ice cream, Pasta*} satisfies the close-family constraint with n=1, because *Salt* is a 1^{st}-cousin of *Ice cream*, and *Ice cream* is a 1^{st}-cousin of *Pasta*. Note that *Salt* and *Pasta*

are only 3rd-cousins. Note that both item constraints and family constraints can be defined using the taxonomy on the ontology, so they are examples of *taxonomical constraints*. It is important to note that item constraints are non-anti monotonic, but both the same-family and close-family constraints are.

The incorporation of an ontology in the mining process, instead of a simple taxonomy, allows for the definition of other constraints that extend the notion of item constraint. Those new constraints are based on the relations among concepts, described in the ontology. Consider some additional predicates over items and itemsets with two or more items (consider an itemset $X \in 2^{\mathcal{I}}$), needed for defining those new constraints. First, two items are *related* if there is a known non-taxonomical relation r (with $r \in \mathcal{R}$) among their classes. Formally,

$$\forall x,y \in \mathcal{I}: related(x,y)=true \Leftrightarrow \exists r \in \mathcal{R}: r(inst^{-1}(x), inst^{-1}(y)).$$

Based on this notion, an itemset is *weakly connected* if all its items are related with at least another item in the set. This is,

$$X \text{ is } weakly\ connected \Leftrightarrow \forall x \in X\ \exists y \in X: x \neq y \wedge related(x,y)$$

The itemset {*Halibut*, *Garlic*, *Lettuce*} is weakly connected in the context of the ontology described on section 3, since there is the relation *Condiment for* from *Spice* to *Main Ingredient*, and another *Enhance flavor* from *Spice* to *Vegetable*. On the contrary, the itemset {*Halibut*, *Port*} is not accepted since there is no relation between *Fish* and *Liquor*. An itemset can either be *softly connected*. This happens when there is a chain of relations among all its items. More specifically

$$X \text{ is } softly\ connected \Leftrightarrow \exists x,y \in X: x \neq y \wedge related(x,y) \wedge X \setminus \{x\} \text{ is } softly\ connected$$

In the context of the same ontology and knowledge base, the itemset {*Halibut*, *Garlic*, *Lettuce*} is also softly connected, because there is a relation from *Halibut* to *Garlic* (*Flavored by*) and from *Garlic* to *Lettuce* (*Enhance flavor*), which means there is a chain of relations among the items.

A third predicate is the strongly connected. An itemset is *strongly connected* if all its items are related to all the other items in the set, which can be stated as follows:

$$X \text{ is } strongly\ connected \Leftrightarrow \forall x,y \in X: x \neq y \Rightarrow related(x,y)$$

In the same context, an itemset that is strongly connected is {*Halibut*, *Bancroft Chardonnay*}, and the combinations among *Bakery* and *Dairy* instances such as {*Eggs*, *Sugar*}. Note that for example, the itemset {*Garlic*, *Halibut*, *Bancroft Chardonnay*} is not strongly connected only because there is no relation from *White wine* and *Spice*.

These three predicates can be used to define new content constraints: *weak*, *soft* and *strong* constraints, respectively. All of them can be categorized as non-taxonomical constraints, since they are based on non-taxonomical relations. From these, only weakly connected is non-anti monotonic.

Beside the existence of taxonomical and non-taxonomical constraints, the *Onto4AR* framework provides another important artifact – the composition of content constraints. Again the composition can be weak or strong.

The *weak composition* of two content constraints, say f and g, results in a new content constraint, $f \circ g$, which only accepts itemsets accepted by at least one of the original constraints. Consider an itemset $X \in 2^{\mathcal{I}}$,

$$f \circ g: 2^{\mathcal{I}} \rightarrow \{true, false\} \text{ with } f \circ g(X)=true \Leftrightarrow f(X)=true \vee g(X)=true$$

On the other hand, the strong composition of two content constraints, say f and g, results in a new content constraint, f o g, which only accepts itemsets accepted by both original constraints. Consider an itemset $X \in 2^I$,

$$f \text{ o } g: 2^I \rightarrow \{true, false\} \text{ with } f \text{ o } g(X)=true \Leftrightarrow f(X)=true \wedge g(X)=true$$

It is important to note that the composition of any pair of the non-taxonomical constraints proposed above, always result in the most restrictive of the constraints. For example, composing weak and soft constraints will result on the last one, since all itemsets accepted by a soft constraint are also accepted by weak constraints. Indeed, the soft constraint is more restrictive than the weak constraint, and the strong constraint is more restrictive than the soft constraint. Interesting compositions can result from combining non-taxonomical with taxonomical constraints. For example, the composition of weak and item constraints would only accept itemsets that have some specific items and, simultaneously, all their items are related with at least another item in the set. In addition to the composition of content constraints, the *Onto4AR* framework allows the definition of other constraints, either as subclasses of proposed ones or by defining new kinds of constraints with different scopes. Examples of the last ones are temporal and structural constraints, used to identify relevant temporal patterns and to identify structured patterns, like frequent sequences or graphs. In order to extend the framework, it is only need to stat the new constraints in accordance to the definitions presented above. In this manner, the categorization of the constraints proposed in this paper, illustrated in Fig. 2, is just a small picture of the potentialities introduced by the *Onto4AR* framework.

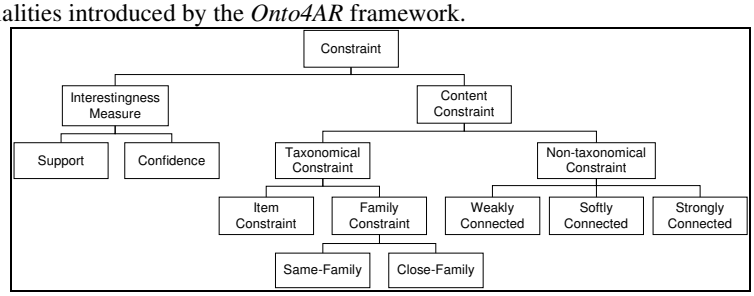

Fig. 2. Categorization of proposed constraints

4.4. Algorithms

As in the rest of pattern mining area, efficient algorithms for identifying accepted patterns is mandatory in this framework. The algorithm for this purpose should be similar to general algorithms for pattern mining, which means that it would filter patterns that satisfy the imposed constraint in an internal pruning step. In this manner, the processing time would be significantly reduced due to the reduction on the explosion of discovered patterns.

It is important to note that this pruning step wouldn't require the navigation over the entire ontology. For each potential pattern (call it *candidate*) we need to verify if the candidate satisfies the constraint, this means, verifying if its items are linked

among them, which only requires to follow the relations among their concepts in the ontology.

The last issue refers to candidate generation, since some of proposed constraints (namely item and weakly-connected constraints) are non-anti monotonic. The simplest solution is to extend frequent k-patterns with frequent items, like it is done in constrained sequential pattern mining [11] and [3]. Fig. 3 presents the pseudo-code for an apriori-based algorithm able to deal with *Onto4AR* constraints.

```
Procedure Onto4AR-apriori (Dataset D, Constraint C)
   L1:={frequent items in D}
   k:=2
   while (Lk≠∅) do
    Ck':=join(Lk, L1, C}           //candidate generation
    Ck:={p∈Ck': p.satisfies(C)     //candidate pruning
    Lk:={frequent patterns in Ck} //support-based pruning
    k:=k+1
return ∪kLk
```

Fig. 3. Algorithm for mining pattern in the *Onto4AR* framework

It is important to note that recent work on sequential pattern mining using more complex algorithms and constraints (deterministic finite and push-down automata) [3] demonstrates experimentally that the reduction on the number of discovered patterns has more impact in the performance, than the verification of the constraint satisfaction for each candidate. Moreover, the *Onto4AR* just use the correspondence between instances and concepts, and the ontology structure – relations among concepts (axioms and instance relations are not used). In this manner, the ontology almost corresponds to an acyclic graph, which makes the verification process similar to the ones referred.

5. Conclusions

The recent advances in the area of knowledge representation makes possible to represent background knowledge, in an effective way, through the construction of ontologies. Since one of the main drawbacks of data mining, in general, and of pattern mining, in particular, is to ignore existing domain knowledge, with those advances, it is time to surpass that feature. This paper presents a framework that incorporates background knowledge in the core of the mining process, by using domain ontologies and by defining a set of constraints above them, which can guide the discovery process. This guidance can be conducted by the expert domain or simply by the user, who establishes the level of constraints to be applied to the process.

The great advantages of the new framework are related to its extensibility, since it is possible to define other constraints based on domain ontologies. In particular, proposed constraints are universal and can be applied to any problem domain, given that there is some corresponding domain ontology.

References

1. Agrawal, R., Imielinsky, T., Swami, A. Mining Association Rules between Sets of Items in Large Databases. In *Proc. ACM SIGMOD Conf. Management of Data*, (1993) 207-216
2. Antunes, C., Oliveira, A., Using Context-Free Grammars to Constrain Apriori-Based Algorithms for Mining Temporal Association Rules. In *Proc. Workshop on Temporal Data Mining* (2002).
3. Antunes, C., Oliveira, A.L., Constraint Relaxations for Discovering Unknown Sequential Patterns. In *Knowledge Discovery in Inductive Databases*, Springer, (2005), 11-32
4. Bayardo, R.J., Agrawal, R., Gunopulos, D., Constraint-Based Rule Mining in Large, Dense Databases. In *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, (1999) 145-154
5. Bayardo, R.J., The Many Roles of Constraints in Data Mining. In *SIGKDD Explorations*, vol. 4, nr. 1 pp. i-ii (2002).
6. Bonchi, F. *Frequent Pattern Queries: Language and Optimizations*. PhD Thesis, Università di Pisa, 2003.
7. Boulicaut, J.-F., Klemettinen, M., Mannila, H., Querying inductive databases: a case study on the MINE RULE operator. In *Proc. Conf. Principles and Practice of Knowledge Discovery*, Springer, (1998). 194-202
8. Češpivová, H., Rauch, J., Vojtěch, S., Kejkula, M. and Tomečková, M., Roles of Medical Ontology in Association Mining CRISP-DM Cycle. In *Proc. Workshop on Knowledge Discovery and Ontologies*. (2004). 1-12
9. Corcho, O., Gómez-Pérez, A., A Roadmap to Ontology Specification Languages. In *Proc. Int'l Conf. Knowledge Engineering and Knowledge Management*, Springer. (2000). 80-96
10. Euler, T., Scholz, M., Using Ontologies in a KDD Workbench. In *Proc. Workshop Knowledge Discovery and Ontologies* (2004). 103-108
11. Garofalakis, M.N., Rastogi, R., Shim, K., SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proc. Very Large Databases Conference*. (1999), 223-234
12. Maedche, A., *Ontology Learning for the Semantic Web*, Kluwer Publishers, (2002).
13. Pei, J., Han, J., Lakshmanan, L.V.S.. Mining Frequent Itemsets with Convertible Constraints. In *Proc. IEEE Int'l Conf. Data Engineering* (2001). 433–332.
14. Pei, J., Han, J., Constrained frequent pattern mining: a pattern-growth view. In *SIGKDD Explorations*, vol. 4, nr. 1. ACM Press, (2002). 31-39
15. Wang, J., Han, J., Pei, J., CLOSET+: Searching the Best Strategies for Mining Frequent Closed Itemsets. *In Proc. Int'l Conf. Knowledge Discovery and Data Mining*. ACM Press, (2003). 236-245
16. Srikant, R., Vu, Q., Agrawal, R. Mining Association Rules with Item Constraints. In *Proc. Int'l Conf. Knowledge Discovery and Data mining*. ACM Press, (1997). 67-73
17. Vanzin, M., Becker, K., Exploiting Knowledge Representation for Pattern Interpretation. In Proc. *Workshop on Knowledge Discovery and Ontologies*. 2004. 61-72