OntoC4S framework: constraint based framework for sequential pattern mining

António Pedro Rodrigues dos Santos Martins Pereira

Técnico Lisboa
antonio.pereira@tecnico.ulisboa.pt

Abstract

Nowadays, data is everywhere and everyone can access it. This data is characterized by its diversification and heterogeneity. So, as consequence, the challenge is to be able of getting useful information of all that data. The interdisciplinary field of data mining appears as a computational process of extracting information from data and transform it into an understandable structure to be applied. Pattern mining is a subfield inside data mining which tries to extract relevant knowledge in the form of patterns from datasets. The final goal of these patterns is to be useful and to help the decision-making process. Constraints are an identified way of getting more interesting patterns and focusing on the pattern mining algorithms to the expectations of the users but until now their importance is very limited by the expression power of the tools used to define them. Ontologies were identified as a new possibility of representing knowledge in a more interesting way. In this work, we developed a ontology based framework for the sequential pattern mining process, the OntoC4S framework, to introduce domain knowledge specified by the user's input. This framework is able to represent sequential and concurrent events, in order to define constraints over sequential data. The goal of this work is to enable the representation of more expressive constraints when compared with other constraints' representation and to obtain a broader way of embody user knowledge while optimizing the current algorithms. The results show that the framework achieves the proposed goals while keeping the performance of unconstraint algorithms. This work is done within the D2PM project where a framework for guiding the pattern mining process is the goal to be accomplished.

Keywords: *Data Mining, Pattern Mining, Constraints, Ontology, OntoC4S framework, SeqD2PrefixGrowth*

I. Introduction

Nowadays data is an abundant and an easily accessible good. In the recent past, the creation, consumption and share of information has increased in an incredible way, accompanied by the technological evolution and the easiest access to that technology for everyone. With this huge amount of new data, lots of challenges and opportunities arise to extract non trivial information that could be useful to the decision making in all levels. With the goal of extracting non-trivial information from data sets, the field of data mining was developed, by combining advanced techniques and methodologies from the fields of machine learning, statistics and databases. The information discovered could become in the form of hidden patterns, unexpected patterns or models governing some classification task. The focus of the algorithms to return results, following the user expectations and the domain that is involved, is an important feature to extract actionable knowledge, i.e. useful information that can be (as far as possible) directly converted into decision-making actions [Cao et al., 2007]. The goal of this domain oriented approach will lead to a better accepted and advantageously useful use in businesses and applications, because the resulting new information will prompt the users to take concrete actions to their advantage in the real world. The state of the art techniques for pattern mining discover a huge amount of

patterns with too few valid domain orientation and useless purpose, further consuming a very large amount of resources. This document introduces the *OntoC4S* framework, a tool created with the goal of fulfilling those issues related to sequential pattern mining. The main goal of this works stands for the definition of a way to define constraints for sequential pattern mining to be adopted in the D2PM framework. These constraints are assigned by the user and mapped to an ontology before being used in the algorithm. This work was done within the *D2PM* project [Antunes, 2011], where a framework for guiding the pattern mining process is the ultimate goal.

II. Related Work

Sequential pattern mining is a field of *pattern mining* with the goal of finding relations between items of sequential events (if there exist any specific order of the occurrences of the items) or itemsets. A sequential pattern mining algorithm mines the sequence database looking for repeating patterns (known as frequent sequences) that can be useful to be used later by end users or managers to find association rules between the different items or events in their collected data with broad applications, including the analysis of customers purchase behavior, web access patterns, scientific experiments, disease treatment, natural disasters and DNA analysis. Given a database D of customer transactions, the problem of mining sequential patterns is to find the maximal sequences among all the sequences that have a certain user's specified minimum support. Each maximal sequence represents a sequential pattern. The discovery of sequential patterns [Agrawal and Srikant, 1995], motivates the problem of finding all sequential patterns following a specific set of constraints, as the support constraint that forces the sequence to be present in the dataset a minimum number of times (user-defined threshold).

Sequential frequent pattern algorithms mainly differ by the way in which candidate sequences are generated and stored, how support is counted and how candidate sequences are tested for frequency. There are three main categories of sequential pattern mining algorithms, namely, apriori-based, pattern-growth and vertical-based. The apriori-based algorithms are known by depending largely on the apriori property, which states that "All nonempty subsets of a frequent itemset must also be frequent", and the use of the Apriori-generate join procedure to generate candidate sequences. Their strategy is characterized by implementing breadth-first search, generate-and-test and multiple scans of the database. Some of the most known apriori-based algorithms are AprioriAll [Agrawal and Srikant, 1995], and GSP [Srikant and Agrawal, 1996]. Pattern-growth category algorithms use depth-first transversal, suffix/prefix growth, and memory-only feature that do not spawn an explosive number of candidate sequences. In this category, algorithms avoid the generation and test of candidates, making use of different approaches to reduce the search space at each step. FreeSpan [Han et al., 2000], PrefixSpan [Han et al., 2001], and GenPrefixSpan [Antunes and Oliveira, 2003] are some of the pattern-growth algorithms.

I. Constrained Sequential Pattern Mining

The discovery of sequential patterns may become a complex task when we are dealing with a large dataset or when the user wants to embody some knowledge in the process. Constraints appear as the most viable way of solving those issues by focusing "*the algorithm on regions of the trade-off curves (or space) known (or believed) to be most promising*" [Bayardo, 2002]. By using constraints, the user assumes the responsibility of choosing which of those aspects are most important for the current task. Constraints represent filters on the data that capture application semantics and allow the users to somehow control the search process and focus the algorithms on what is really interesting. The most used constraint in sequen-

tial pattern mining is the user-specified minimum support [Agrawal and Srikant, 1995]. The support of a subsequence is the fraction of sequences in the dataset where the subsequence is present at. Support could also be not considered as a constraint because it is the basis measure for the pattern mining process. The algorithms that just make use of the support, return a very large number of sequential patterns, most of them completely uninteresting for the user. This lack of user-controlled focus suffers from two major drawbacks, as stated in [Garofalakis et al., 1999], disproportionate computational cost of selective users and overwhelming volume of potentially useless sequences and rules. To resolve these issues many efforts have been made to push user's domain knowledge and expectation inside pattern mining algorithms. Some important algorithms that make use of constraints are *SPIRIT* [Garofalakis et al., 1999] and *PrefixGrowth* algorithm [Pei et al., 2002]. The *SPIRIT* algorithm follows the structure of apriori strategy and make use of regular expressions for the discovering of sequential patterns. *PrefixGrowth* follows the strategy adopted by the *PrefixSpan* and incorporate prefix-monotone constraints in the mining process.

II. Limitations of using Regular Languages

The use of regular languages to restrict the returned result set in the mining process over sequences was the first approach to embody knowledge and, consequently, to obtain more focused and interesting patterns. Indeed, it is only viable in a situation where the knowledge to be contained in the constraint is relatively simple. But If we want to increase the complexity of the DFA, and consequently, define a richer constraint, we easily get a larger and more confusing schema.

Summing, the DFA is not a valid tool when there is interest in embodying knowledge and when used the DFA gets complex to define and understand. This increasing of complexity could have more consequences as the decrease of efficiency of the algorithm running. A different way to represent constraints is using a context-free language. This approach introduces two challenges: how to effectively manipulate the pushdown stack, and the best way to deal with the non-deterministic property of the languages. In order to enable the use of context-free languages the ePDA [Antunes, 2005] was created. The ePDA is an extension of the notion of PDA to be used in the *sequential mining* process. *GenPrefixGrowth* [Antunes, 2005] is an algorithm created to make use of ePDA and is based in the *GenPrefixSpan* algorithm. The definition of this kind of constraints and the *GenPrefixGrowth* algorithm lead to some interesting results, as stated by the authors: the incorporation of constraints in the sequential pattern mining leads to a more efficient process when compared with a post-processing of the result set; and the use of context-free languages doesn't invalidate all the issues showed before.

III. Open Issues

The incorporation of domain knowledge in data mining has been identified as one of the most important challenges [YANG and WU, 2006], due to the possibility to guide the algorithm through the discovery of more focused and, consequently, returning of more interesting results, besides the representation of domain semantics and user expectations. We can define this embody of knowledge in the process as the need from two points of view, the academic: which wants to optimize the performance and introduce generalization in the process; and the business view which wants to be able to incorporate complex knowledge environment in the algorithms through constraints that can answer in time to the real needs and may guide the decision-making process [Cao et al., 2007].

Using constraints to incorporate domain knowledge in (sequential) pattern data mining, besides of the advantages stated before, re-

duce the set of resulted patterns (and possibly the search space in the algorithm), minimizing two problems of data mining field. One premise that has to be present when defining constraints is not to push too restrictive constraints that could limit the patterns discovery to a simple hypothesis testing approach. A more expressive way of representing domain knowledge is made by graphical models. With these models, users and experts are able to represent the domain with concepts hierarchies and relations. As an example of these models there are the taxonomies, which model an is-a relation between concepts. By using these models to define constraints in data mining process, they may lead to more interesting results that could span different levels of the taxonomy in the hierarchy of concepts, enabling the generalization of results and the prune of redundant results [Silva and Antunes, 2014]. An ontology is an explicit specification of concepts and relationships that can exist between them. Ontologies are content theories about the objects, their properties and the relations that are possible in a specified domain of knowledge, formalizing the system of knowledge representation for that domain [Chandrasekaran et al., 1999]. It is an agreement of an explicit specification of a shared conceptualization of a domain, not just a representation vocabulary. Consequently, similar to constrained data mining, we can also define a set of constraints based on the characteristics of an ontology that can then be used by the presented constrained approaches [Antunes, 2008].

III. *OntoC4S* framework

To achieve the goal of incorporating domain knowledge and the use of ontologies as guiders of the pattern discovery algorithms in data mining, we developed a framework composed by an ontology (*OntoC4S*) that maps constraints over sequences and an algorithm (*SeqD2PrefixGrowth*) that uses the ontology to create rules that respect the Constraint sequence defined and return frequent patterns over that rules. The goal of this framework is to incorporate the constraints defined in the ontology inside the pattern mining process and limit the return set by the interest of the user.

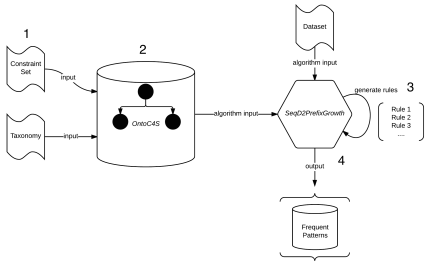

Figure 1: *OntoC4S framework overview.*

In the figure 2 it is possible to observe the main flow of the framework since the definition of the constraint sequence to the extraction of the final set. As indicated, the first phase consists in the definition of the constraint sequence and the taxonomy, followed by the read of these inputs to the *OntoC4S* ontology. After mapping from the input to the ontology, the algorithm *SeqD2PrefixGrowth* is responsible for the interpretation of the ontology and the translation of the constraint sequence to the respective rules that will be used to force the frequent patterns of the dataset through the algorithm.

For the user to specify the constraint sequence and the taxonomy embodied with the knowledge he wants to insert in the framework, he will need to map that information in a semi-structured representation, by the means of a *XML* file. The schemas for both elements (the ontology and the taxonomy) were designed to allow an easy approach to define knowledge.

I. *OntoC4S* ontology

The *OntoC4S* is an ontology to represent constraints through the specification of sequential and concurrent events and their relations. It is integrated in the *D2PM* framework [Antunes, 2011], a framework that has the goal of supporting the process of pattern mining with the

use of domain knowledge, and encloses the full range of pattern mining methods, from transactional to sequential and structured pattern mining. To create the *OntoC4S* ontology, we analyzed some ontologies and languages independent of the application domain they were made but following the features we referenced before. The Process Specification Language [Schlenoff et al., 1999] is an ontology with the goal of providing a domain and tool neutral representation (formally describing concepts, along with their properties and relationships) of manufacturing processes and it was identified as the a promising instance to be took as reference in the design of the *OntoC4S* ontology. In this ontology, the constraints are represented in a sequence. This sequence starts by specifying the initial configurations,i.e. the initial gap allowed to the first constraint of the sequence. After that, the constraints are defined and between a pair of constraints the ontology has an entity which define what is the type of relation between them. A pair of constraints could be sequential or concurrent. When they are sequential, i.e. in the context of a sequential pattern they belong to the same itemset, a concurrent relation entity is set, and when they are sequential a sequential entity is instantiate and a gap could be define. This gap states the maximum that could exist between the consecutive constraints. The properties of each type of constraint depends on the relevant information that could be defined related to it. As an example, consider the *exist* constraint which states that an item exists in a pattern sequence. The relevant information to define here is the item that we want to exist. In the case of a *precedence* constraint (which affirms that an item should come before other item) the interesting information to define is the items and the maximum gap allowed between them.

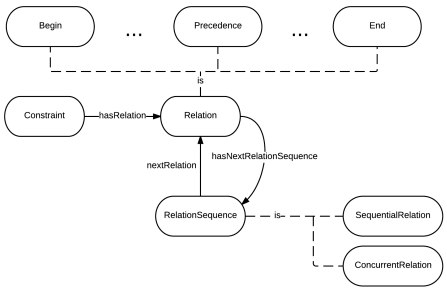

Figure 2: *OntoC4S architecture.*

To illustrate the use of this ontology, let's consider an example in the healthcare domain. Consider the nursery protocol that stands for the injection of an oncology patient. The protocol is constituted by 5 activities: tourniquet application; disinfectant application; disinfectant dry; catheter insertion; and patient injection. If we understand the activities as being sequential the UML (Unified Modeling Language, version 2) in the figure 3 is an illustration of the protocol.

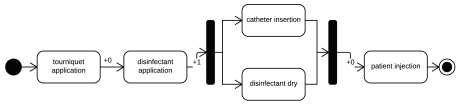

Figure 3: *Sequence of a nursery protocol with concurrency and the gaps between the activities.*

As can be seen, there are two concurrent constraints, i.e. item catheter insertion is to happen when the disinfectant dry is executed (and vice-versa), belong to the same itemset/transaction. The representation using the *OntoC4S* ontology is showed in the figure 4.

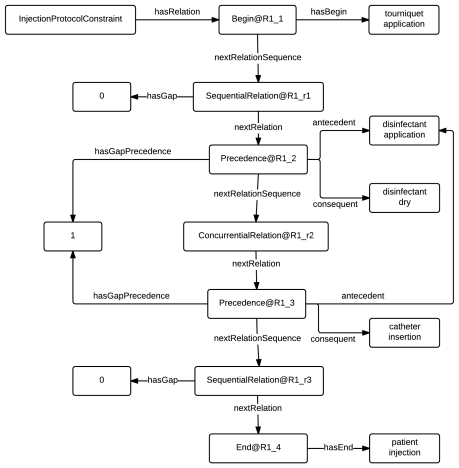

Figure 4: *Example of the Figure 3 through the ontology ontoC4S.*

The sequence starts with the "InjectionProtocolConstraint" with gap value of 0 (omitted) and then the constraints are defined according to their order. Between a pair of constraints we could find the specification of the kind of relation (sequential or concurrential). As an extension to the ontology and the flexibility of the *sequential pattern mining* algorithm, a taxonomy is integrated. In this taxonomy is possible to define items as composed with other items (creating a tree of items). In this way, it is possible to define constraints with item values that could be broader than a specific item.

II. *SeqD2PrefixGrowth* algorithm

The *SeqD2PrefixGrowth* algorithm was created with the goal of allowing the incorporation of constraints provided by an ontology in the process of sequential pattern mining. The ontology used to provide the constraints is the *OntoC4S* ontology. This algorithm is based on the *GenPrefixGrowth* algorithm [Antunes, 2005], an extension of the *GenPrefixSpan* [Antunes and Oliveira, 2003] algorithm. The algorithm starts after the ontology finishes the reading of the inputs (constraint sequence XML file and the taxonomy XML file) and represents the knowledge in its representation model. Only when the algorithm takes into scene the dataset is taken in account. The algorithm has 2 different phases:

- **first phase:** the read of the ontology's knowledge and the representation of it in a object oriented structure (using Java object oriented programming language);
- **second phase:** sequence pattern mining algorithm execution.

II.1 Algorithm's first phase: ontology processing

The first phase is responsible for the ontology's constraint sequence information and taxonomy read followed by the instantiation in a object oriented class structure designed to accommodate it. The ontology is processed different from the taxonomy. While this last is mapped in a object based element structure similar to the ontology entities used to define it, the information related to the constraint sequence is mapped into rules. Each constraint is represent by one or more rules depending of the constraint's kind. A rule is a concept that affirms something about the existence of an item (in the limit all we need was an *exist* kind constraint to represent all the constraints) and is the primitive concept used to define a constraint. A rule is constituted by the needed attributes to express all the knowledge related to an item in this context and it is represent by its own *Java* class. The attributes that characterizes a rule are the same independent of the kind of constraint it will represent. In the next list they are identified and explained:

- **Value:** identifies the item that the frequent pattern should have to respect the constraint (or part of it);
- **RuleOrder:** the number of the rule. This number represents the order in which the rule were defined;
- **ConstraintOrder:** the number of the constraint being processed. This number could be different of the rule order due to the possibility of a constraint to be represent by more than one rule;
- **Gap:** the maximum gap (or itemsets) allowed to the discovery of a situation

which respects the rule;
- **isParallel:** a binary possible value that indicated if this rule should be parallel with other previous rule (in order to be possible to represent parallel constraints), i.e. must be present in the same itemset.

II.2 Algorithm's second phase: sequence pattern mining process execution

This is the phase for the sequential pattern mining algorithm to run. As it is easy to understand, the phases before occurred first because the output they return are directly or indirectly needed for the algorithm to run, i.e. the taxonomy and the rules. Besides these inputs to the algorithm, as a follower of the *GenPrefixSpan* algorithm it will receive the maximum distance allowed when counting the elements for the generation of candidate patterns, the minimum support for a sequence to be considered a frequent pattern (as in all pattern mining algorithms), and the data set itself from where the frequent patterns will be extracted from.

The first step is the discovery of the *1*-itemsets. To do it, the algorithm counts each of the elements in the alphabet of the dataset in each sequence. At most, the same element counts by appearing in a sequence independent of how many times in the sequence. Then, by the order of the support of the elements (to be more efficient), the algorithm validates that the *1*-itemset candidate pattern has a support greater or equal to the minimum support value (for a sequence to be considered frequent pattern in a unconstraint algorithm, the elements that build the sequence just must have a support number above or equal to the minimum support value) and if this first item respects the rules (in this case the first rule). If the *1*-itemset validates the rule then other elements could be appended to this itemset in order to verify if they are frequent patterns, if not the itemset is discarded. When an item is identified as being frequent and is created an frequent itemset with it, the algorithm builds a projected database (α-projected database) with this item as the prefix. Conceptually, a projected database is a subset of the original dataset with the sequences that have this frequent itemset as a prefix, and these sequences are truncated till this prefixes in order to be more efficient to search for new items to append. Then the algorithm will enter in a recursive method in order to increment the size of the patterns. This algorithm follows a depth-first search, i.e. will increase an frequent pattern till that sequence couldn't be more increased. Each iteration of this method code is responsible for an increment of one item in the actual frequent patters. This cycle needs to be in account some contextual situations, the α that defines the prefix from where the new element will be appended, the size of this α (number of items which build the actual pattern), the elements of the alphabet in order to know what are the element it will search and append, and the *projected database* of this prefix (built before).

Other important phase is the verification of the compliance of the candidate itemsets over the rules created from the ontology. When a new candidate is received, it's verified with the actual rule. To compare a rule with the candidate item, the algorithm verifies if the itemset's item order in the sequence is inside the allowed gap, and if it is, verifies if the item is equal or belong to the item (in case of a composed item) indicated by the rule. When the item doesn't verify the rule, the candidate is automatically discarded. When the candidate respects the rule it could accomplish the rule or be still in the allowed gap before the items' rule must be present. In the case of just being in the allowed gap, the rule maintains the same and the gap is updated. If the candidate item is equal to the rule's item or belongs to the same taxonomy tree, a new rule is loaded from the processed rules. Each pattern verification is accompanied by a state that identifies uniquely the context of that pattern in that specific moment. The atributes of the state needed to identify the context are:

- **Max allowed itemset:** it is the sum of the last passed rule itemset with the max-

imum gap allowed in the next rule. The actual candidate item's itemset must be have the itemset order less or equal to the this number in order to be accepted.

- **Rule order:** the order number in which the rule were defined;
- **Restriction order :** the order of the constraint being processed and which this rule belongs to. This number could be different of the rule order due to the possibility of a constraint to be represent by more than one rule;
- **Current itemset:** the order number of the current item's itemset;
- **Start rule itemset order:** the first itemset to consider when verifying the constraint (when items of an accepted item's itemset are being caught);
- **Item found in itemset:** identifies the item's itemset order number of the accepted;
- **Validation state:** describe the state of the validation in the actual context. The possible values are: no rules, inside, pass, all passed.

In overview, the mining process is composed by some concepts beyond the main procedure of the process. The algorithm entity is the concept class that coordinates the process but the features don't just stand on it. Other concepts were created in order to enable the implementation of the algorithm features in order to ease the management and flexibility of the process. The concept of constraint in an class structure enables the management of the validation and verification of the constraints and the discovery of the projected databases and the count of the elements in the same projected databases or/and initial dataset. The governance of the rules and the rules sequence is responsibility of a concept called "tree". The rule is the concept with responsibilities in verifying the conformity of the input item with the rule according to the respect of the gap limit or the familiarity between the rule's item and the input one. This concept is called from the manager concept (the "tree"). As stated before, the mining pattern sequence has a context represented by the concept "state" explained above. The concept which represents the actual pattern is the "sequence". The meaning of this concept is the same of its name and is composed by itemsets. Itemsets have items which represents the concept that is being tested each time over the rule.

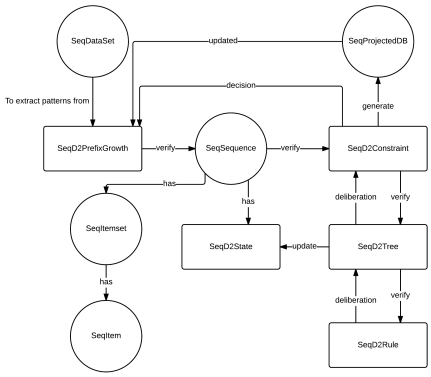

Figure 5: *SeqD2PrefixGrowth algorithm mining process overview.*

IV. Evaluation

Our goal in this section is to measure and understand the impact of those characteristics in the performance of the algorithms (execution's s time and consumed memory). In order to do that, we compare the performance of *GenPrefixSpan* and *SPaRSe* with the performance of *SeqD2PrefixGrowth* algorithm. The basic methods used by the different algorithms were the same (basic operations on sequences, support verification, etc) which means that the comparisons are meaningful and repeatable. The datasets were maintained in main memory during the processing, avoiding hard disk accesses. First we had the goal of this analyzing the behavior pattern of the algorithm over different gap values (gap related to the counting of the elements). We tested the algorithm's performance time with the gap value of 1, 2 and 3. As it was expected, the algorithm's execution time increases as the gap used in the count phase increases. It is now important to understand where the algorithm wastes more time.

Figure 6: *Execution time by algorithm mining phases.*

Our results show that the mining process defines the execution time because all other time consumed in the other phases are irrelevant when compared. We decided to drill-down and to understand how the mining process divide its time. The mining process is constituted by 3 main phases: the creation of the projected databases, the candidates generation, and the constraints verification. In average, the generation of the candidates reveal to be the most time waster, consuming approx. 61% of the time. Other important aspect was to verify the performance of the *SeqD2PrefixGrowth* when compared with unconstraint algorithms, in order to verify if the algorithm didn't lose the features of those algorithms. The results showed us that our algorithm consume five times more memory although the execution time be very similar. This difference is due to the fact that for each candidate pattern there is a state. Future improvements can be done to improve the memory consumed.

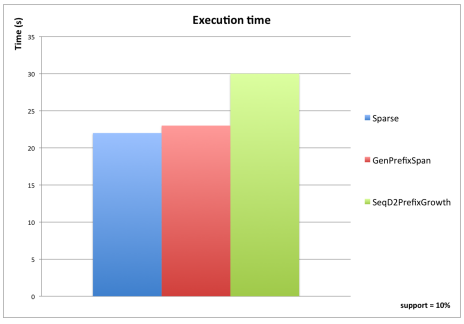

Figure 7: *Execution time comparision with unconstraint algorithms.*

Summing, they have similar performances if we have in mind the features of each algorithm, besides the sizable difference in the memory requirements. We need to be conscious that there is a trade-off when we are adding complexity to an algorithm to add new features as the verification of constraints in patterns.

V. Conclusions and Future work

In this work, we studied the sequential pattern mining field and algorithm, with more focus on the discovery of patterns with the attention of using constraints, and presented some issues that were identified with the importance and impact of the constraints in the pattern mining process. We understood that there are strong limitations in the expression power of the constraints used until now (as pointed with the limitation of regular languages) which, consequently, affect the returned results when compared with the user expectations. This constraints limitation are due mainly to the power of the tools that are used to represent them and, so, ontologies were proposed as a better way of representing them [Antunes, 2008]. The results of this work showed that while keeping the returned sequences framed in the user defined in- put (ontology and taxonomy) and domain knowledge, keeps the features and comparative performance of the most efficient unconstraint algorithms (as the *GenPrefixSpan* [Antunes and Oliveira, 2003]). This ontology is part of the OntoC4S framework where is present the OntoC4S ontology, the algorithm created in this work, *SeqD2PrefixGrowth*, and designed inputs to enable the use of the framework to the intended goals. As showed in this document, the framework enables the use of constraints over the sequential mining process, with the possibility of these constraints to be over sequential or parallel events, allow the existence of gap between the elements in the constraint and the specification of complex constraints (based in more simple constraints).

Future work could be done to increase the efficiency and the effectiveness of the *SeqD2PrefixGrowth* algorithm and the *OntoC4S*

framework in general. The application of the framework in a concrete study case could generate some interesting information and more action oriented when compared with unconstraint algorithms and the actuals constraint algorithms. To allow a more flexible constraint definition, the framework can be changed to deal with a tree of constraints instead of a sequence of constraints, define other kind of constraints (cardinality constraints), and create a visual tool to define the constraints and the taxonomy to ease the process of defining constraints and relations of items.

References

Agrawal, R. and Srikant, R. (1995). Mining Sequential Patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society.

Antunes, C. (2005). *Pattern Mining over Nominal Event Sequences using Constraint Relaxations*. PhD thesis, Universidade Técnica de Lisboa.

Antunes, C. (2008). An ontology-based framework for mining patterns in the presence of background knowledge. In *Int'l Conf. on Advanced Intelligence, Beijing, China*, pages 163–168.

Antunes, C. (2011). D2PM: Domain Driven Pattern Mining. Technical report, project report, Tech. Report 1530, IST, Lisboa.

Antunes, C. and Oliveira, A. L. (2003). Generalization of Pattern-growth Methods for Sequential Pattern Mining with Gap Constraints. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM'03, pages 239–251, Berlin, Heidelberg. Springer-Verlag.

Cao, L., Luo, D., and Zhang, C. (2007). Knowledge Actionability: Satisfying Technical and Business Interestingness. *Int. J. Bus. Intell. Data Min.*, 2(4):496–514.

Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999). What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems*, 14(1):20–26.

Garofalakis, M. N., Rastogi, R., and Shim, K. (1999). SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 223–234, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000). FreeSpan: Frequent Pattern-projected Sequential Pattern Mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 355–359, New York, NY, USA. ACM.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224.

Pei, J., Han, J., and Wang, W. (2002). Mining Sequential Patterns with Constraints in Large Databases. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 18–25, New York, NY, USA. ACM.

Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Road, T. C., Inc, S., Lubell, J., and Lee, J. (1999). The Process Specification Language (PSL) Overview and Version 1.0 Specification.

Silva, A. and Antunes, C. (2014). Domain Knowledge and Data Mining. Technical report, IST, Lisboa.

Srikant, R. and Agrawal, R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pages 3–17, London, UK, UK. Springer-Verlag.